2002

Javaserver page, Java servlet and JavaBean technology: Online real estate company

Kevin Tzu-Jung Chen

JAVASERVER PAGE, JAVA SERVLET AND JAVABEAN TECHNOLOGY: ONLINE REAL ESTATE COMPANY

A Project

Presented to the

Faculty of

California State University,

San Bernardino

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

in

Computer Science

---

by

Kevin Tzu-Jung Chen

June 2002

JAVASERVER PAGE, JAVA SERVLET AND JAVABEAN TECHNOLOGY: ONLINE REAL ESTATE COMPANY

A Project

Presented to the

Faculty of

California State University,

San Bernardino

---

by

Kevin Tzu-Jung Chen

June 2002

Approved by:

[redacted]

Dr. David Turner, Chair, Computer Science — Date 6/5/2002

[redacted]

Dr. Richard Botting

[redacted]

Dr. Ernesto Gomez

ABSTRACT

In this project, I will simulate an online real estate company using JSP, Java Servlet and JavaBean technology. It is not as complicated as an online company in the real world, but it includes all the basic functions that an online company needs. This online real estate company - SweetHome - is a brokerage agent that provides easy online solutions for house buyers and sellers. Properties information will be stored in a Relational Database for the users to search. The customer can register as a new member, or even only just as a guest on line.

I designed the system to use a 3-tier architecture. In the client tier, users can access various information and services through a web browser. In the Application server tier, I use JSP and Java Servlets. In the business logic tier, I use JavaBeans to connect to the database by JDBC.

JSP (JavaServer Page) is an extension of the Java servlet technology from Sun that provides a simple programming vehicle for displaying dynamic content on a Web page. Java Servlet is a Java application that

runs in a Web server or application server and provides server-side processing, typically to access a database or perform e-commerce processing.

JavaBean is a reusable software component that can be visually manipulated in builder tools. I use JavaBeans in the web application tier to communicate with the MySQL database in the database server tier.

The goal of this project is to use JSP, Java Servlet and JavaBean technologies to design and implement the SweetHome system.

ACKNOWLEDGMENTS

I would like to thank Dr. Richard Botting and Dr. Ernesto Gomez for being my project's committee members. They have provided me with many precious suggestions and instructions.

Also, extreme thanks to my advisor Dr. David Turner. Dr. Turner has given me many ideas, recommendations, and help throughout the project design and implementation. He is not only a good teacher in the computer field but also a good friend who always reminds me what I should do next with patience.

Finally, profound thanks to my parents and my sister; without their support and encouragement, I could not have had this opportunity to pursue higher education in America. They are the most important people in my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE

INTRODUCTION

1.1 Introduction

Web based application is becoming more and more popular these days. Many dot com companies such as Amazon, eBay, and other online companies provide their business services through the Internet. Customers can log into a company's web site to purchase goods and services. Online companies can provide their information and services without opening a real retail store.

In this project, I will simulate an online real estate company. This online real estate company - SweetHome - is a brokerage agent that buys and sells properties, especially houses, in California. SweetHome provides basic functions for those who want to find appropriate properties or brokers through the Internet. Customers can search property in the SweetHome website. They can also choose their brokers depend on what language the customers speak, as well as the brokers' experience, and sale records etc. After registration, if customers want to buy or sell

their houses, they can make an appointment with their brokers.

The most two popular platforms for web application development are Microsoft .NET and Java J2EE. I choose the Java platform in this project is based on the following reasons:

First, it works on multiple platforms; once you finish the web application, you can run it both on Linux or Windows.

Second, it is an open source; you can download the J2SE or J2EE from Sun's website for free.

Third, it is object oriented; compared to C++ and Visual Basic, Java is an object oriented language.

Thus, I chose to use Java related technologies to implement this system.

1.2 Scope

SweetHome will provide services and information for customers who want to buy or sell their houses. This online real estate company could support two kinds of scenarios.

First, the customers are property buyers. The application will provide the interfaces so that the

buyers can search the properties on the market, find the appropriate office and broker, register them on the web site, and make an appointment with their broker.

Second, the customers are property sellers. The scenario for the sellers will be the same as the one for buyers except for searching for properties.

The application starts with customers logging into the web site and ends with customers logging off the web site.

1.3 Definitions

Table 1. Definitions

Java	An object oriented language developed by Sun Microsystems. Java programs are capable of running on most popular computer platforms without the need for recompilation.
JSP	JavaServer Page, an extension to the Java servlet technology from Sun that provides a simple programming vehicle for displaying dynamic content on a Web page.
Java Servlet	A Java application that runs in a Web server or application server and provides server-side processing, typically to access a database or perform e-commerce processing.

JavaBean	A component architecture for the Java programming language, developed initially by Sun, but now available from several other vendors. JavaBeans components are called "beans."
JavaScript	A scripting language that is widely supported in Web browsers and other Web tools. It adds interactive functions to HTML pages, which are otherwise static.
JDBC	Java Database Connectivity, a programming interface that lets Java applications access a database via the SQL language.
JDK	Java Development Kit, a free Sun Microsystems product which provides the environment required for programming in Java. The JDK is available for a variety of platforms, such as Sun Solaris, Microsoft Windows and Linux.
J2EE	Java 2 Platform, Enterprise Edition. Sun's Java platform for multi-tier server - oriented enterprise applications.
EJB	Enterprise JavaBeans, server-side component architecture for writing reusable business logic and portable enterprise applications. EJB is the basis of Sun's J2EE.
HTML	Hyper Text Markup Language, a document format used on the World-Wide Web.
HTTP	Hyper Text Transfer Protocol, the protocol that defines how messages are formatted and transmitted on the World Wide Web
IEEE	Institute of Electrical and Electronics Engineers. The world's largest technical professional society, based in the USA. Founded in 1884 by a handful of

	practitioners of the new electrical engineering discipline, today's Institute has more than 320,000 members who participate in its activities in 147 countries.
OS	Operating System. The low-level software which handles the interface to peripheral hardware, schedules tasks, allocates storage, and presents a default interface to the user when no application program is running.
SQL	Structured Query Language. A standard language that provides controlled access to databases.
Browser	A program capable of retrieving HTML documents that includes references to images and Java byte code and rendering it into a user-readable document.
Server	A program which provides some service to other (client) programs. The connection between client and server is normally by means of message passing, often over a network, and uses some protocol to encode the client's requests and the server's responses.
Client	A computer system or process that requests a service of another computer system or process (a "server") using some kind of protocol and accepts the server's responses. A client is part of client-server software architecture.
API	Application Programming Interface. The interface by which an application program accesses operating system and other services. An API is defined at source code level and provides a level of abstraction between the application and the kernel (or other privileged

	utilities) to ensure the portability of the code.
GUI	Graphical User Interface. The graphical representation of physical or pseudo-physical objects (such as buttons, trees, and lists) that allow the user to direct the flow of the program through the use of a mouse or other pointing devices.
URL	Universal Resource Locator, a standard way of specifying the location of an object, typically a web page, on the Internet.

CHAPTER TWO

SYSTEM ARCHITECTURE

2.1 The 3-tier Architecture

The infrastructure of SweetHome is a 3-tier architecture (See Figure 1), using web technologies including multi-platform browsers, JavaBean, HTTP servers, HTML documents, JSP, Java Servlet, and JDBC access to Relational Databases.

I considered several other architectures available for this project, such as the 2-tier architecture. The 2-tier architecture is the traditional idea of a "client-server" system. Usually, this architecture combines both presentation logic and business logic together in one tier. The server provides persistent storage via a database.

The 3-tier architecture separates out the presentation logic from the business logic (which resides on a middle-tier system). Considering the scalability and flexibility for this business, I decide to set up a 3-tier web deployed architecture. There are many advantages of using a 3-tier architecture with the web.

First, The 2 -tier architecture does not have an application server tier. Business-objects that implement the business rules "live" here, and are available to the client-tier. This tier protects the data from direct access by the clients.

Second, as the Internet and Intranet become more and more powerful and widely-used in business and daily life, the World Wide Web is the perfect place to advertise the products.

Third, the cost of setting up a site is reasonable and affordable ($70 to obtain a domain name for first 2 years and $35 per year after 2 years).

Finally, a system deployed through the World Wide Web can avoid the hassles of client-software installation and upgrades. Clients simply need access to a browser, which can be downloaded free of charge from the web. The security, scalability, and platform independence are the main reasons why I rejected the 2-tier architecture.

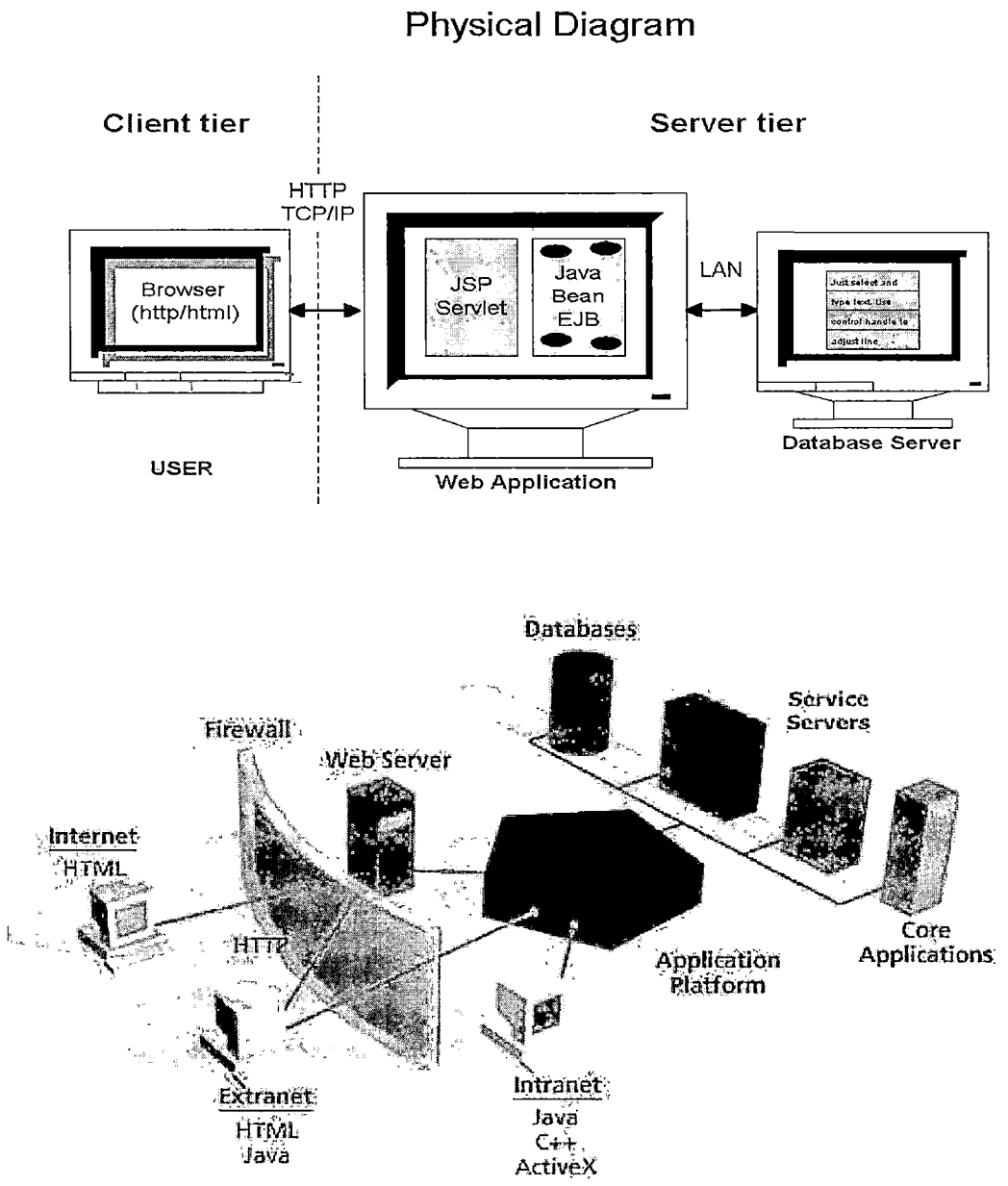


Figure 1. Physical Diagram

2.2 Client Tier

The web browser is the minimalist client that sends users' requests and interprets information it receives from a server, and displays it graphically to a user. The client is simply there to interpret the server's commands and render the contents of a HTML page to the user. Web Browsers are primary interpreters of HTML syntax. The browser executes the HTML commands to properly display text and images on a specific GUI platform. Users navigate from one page to another using the embedded hypertext links.

In this project, I needed to develop a website that contained the main page and a pop-up window for mobile users to login to the Internet.

Operating System

This is a client platform independent system. The client can use any operating systems, such as Windows 95/98/NT/2000/XP workstation, Mac OS, OS/2, UNIX, Linux etc. Netscape Navigator 3.0 or higher or Microsoft's Internet Explorer 3.x or higher is required to view the HTML documents and web forms on the Web Server.

2.3 Application Server Tier

A World Wide Web server is simply a program that answers requests for documents from World Wide Web clients over the Internet. All World Wide Web servers use a language, or protocol to communicate with web clients called the Hyper Text Transfer Protocol. This is where the http in a web URL comes from. All types of data can be exchanged using this protocol including HTML, graphics, sound and video. Web clients convert open URL commands into HTTP GET requests.

Considering the performance issue, which is basically the waiting time, and also about the budget issue, which has to be pressed as low as possible, this project was designed to run in Jakarta Tomcat 4.0.1 web server.

SweetHome web site is not a very heavy site that provides thousands of services to people. This is different from sites such as yahoo.com, which has yahoo games, yahoo maps, online shopping, chatting, messenger, and so on. Tomcat web server, which is the product of Jakarta project from Apache, is offered for free by downloading directly from the Apache website,

http://jakarta.apache.org/. Moreover, Tomcat also supports Servlets and JSP programming, which are the main technology used in this project.

2.4 Database Server Tier

Of course, every system must have a place to store and retrieve massive loads of data. I decided to use MySQL 3.23 as my DBMS server.

MySQL is free software and there are different versions that can run on Windows, MacOS, Solaris, HP-UX, AIX and Linux. It supports APIs for C, C++, Eiffel, Java, Perl, PHP, Python, and Tcl. It provides very fast joins using an optimized one-sweep multi-join. You can connect to MySQL by running a middle-tier separate application server that has a MySQL driver.

2.5 Middleware Services

Middleware starts with the API set on the client side that is used to invoke a service and covers the transmission of the request over the network and the resulting response. In N-tier environments, middleware must provide a platform for running server-side

components, balancing their loads, managing the integrity of transactions, maintaining high-availability, and securing the environment. Here are two general middleware systems:

Object-Specific Middleware - The Hypertext Transfer Protocol

The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. It is a generic, stateless, object-oriented protocol that can be used for many tasks, such as name servers and distributed object management systems, through extension of its request methods. A feature of HTTP is the typing and negotiation of data representation, allowing systems to be built independently of the data being transferred. HTTP has been in use by the World-Wide Web global information initiative since 1990.

Database-Specific Middleware - Java Database Connectivity

JavaSoft's Java Database Connectivity (JDBC) is an API that lets you access virtually any tabular data source from the Java programming language. It provides cross-DBMS connectivity to a wide range of SQL

databases (See Figure 2). It makes Java codes DBMS-independent. Almost all of main database vendors provide their own JDBC drivers in multi-platform.

Therefore, at most time, the database operation program written in java with JDBC is easy to be migrated to many kinds of platforms without changing the code.

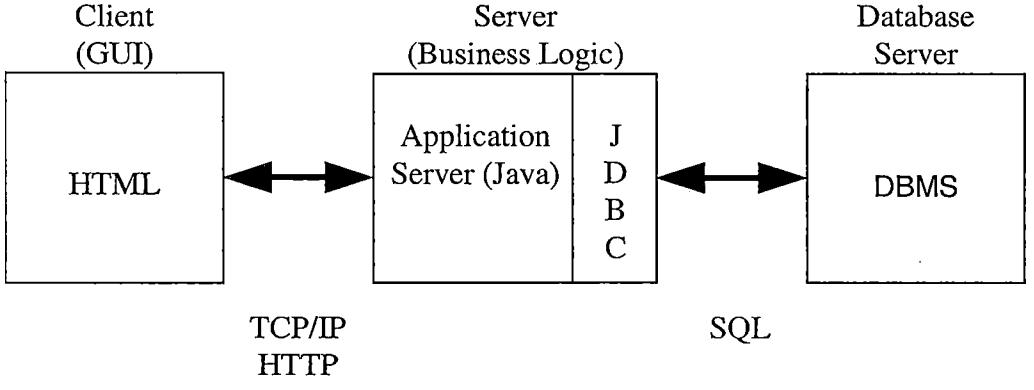


Figure 2. Java Database Connectivity Diagram

CHAPTER THREE

JAVASERVER PAGE, JAVA SERVLET AND JAVABEAN

3.1 JavaServer Page versus Active Server Page

JavaServer Pages (JSP) and Microsoft Active Server Pages (ASP) technologies have many similarities. Both provide a simplified, fast way to create web pages that display dynamically generated content. But they also differ significantly in some ways. Here are some main strengths of JSP as compared to ASP:

Portability

JSP is being developed by Sun Microsystems and is designed to be both platform and server independent (See Table 2.).

Therefore, it has "Write Once, Run Anywhere" capability. In contrast, ASP is purely a Microsoft based technology deployed primarily on Windows servers. JSP technology was designed to support numerous servers, browsers and tools. For example, Apache web server, which hosts more than 70/% of the

websites worldwide, will now fully support the JSP technology. So, you do not have to worry about your OS being Windows or Linux.

Reusability

Most JSP pages rely on reusable, cross-platform components (See Table 2.) to perform the more complex processing required of the application. Developers can instantiate JavaBeans components, set or retrieve bean attributes and perform other functions that are otherwise more difficult and time-consuming to code.

They also can share and exchange components that perform common operations, or make them available to larger customer communities. The component-based approach speeds overall development and lets organizations leverage their existing expertise and development efforts for optimal results.

Performance

The JSP page is compiled into a Java Servlet class and remains in server memory after it has been called for the first time, so subsequent calls to the page have faster response time whereas in ASP the page needs to be recompiled for every request.

Custom Tag Libraries

The JSP technology is extensible through the development of customized tag libraries. The web page developers can create custom tag libraries, so page authors can access more functionality using XML-like tags and depend less on scripting. With custom tags, developers can shield page authors from the complexities of page creation logic and extend key functions to a broader range of authors.

Table 2. Active Server Page versus JavaServer Page

	ASP Technology	JSP Technology
Web Server	Microsoft IIS or Personal Web Server	Any Web server, including Apache, Netscape, and IIS
Platforms	Microsoft Windows	Most popular platforms, including the Solaris, Microsoft Windows, Mac OS, Linux, and other UNIX platform implementations
Reusable, Cross-Platform Components	No	JavaBeans, Enterprise JavaBeans, custom JSP tags
Compiles	Every time	First time
Customizable Tags	No	Yes

3.2 JavaServer Page

Java Server Pages (JSP) is a server-side scripting language which is based on the Servlet framework and allows combining of HTML text with Java source code in the same document. The suffix traditionally ends with .jsp to indicate to the web server that the file is a JSP file. JSP is a *server side technology* - you can not do any client side validation with it.

JSP sits on top of a Java servlets model and makes working with HTML easier. It allows developers to mix static HTML content with server-side scripting to produce dynamic output. By default, JSP uses Java as its scripting language; however, the specification allows other languages to be used.

To offer the best of both worlds - a robust web application platform and a simple, easy-to-use language and tool set - JSP provides a number of server-side tags that allow developers to perform most dynamic content operations without ever writing a single line of Java code. So developers who are only familiar with scripting, or even those who are simply HTML designers,

can use JSP tags for generating simple output without having to learn Java. Advanced scripters or Java developers can also use the tags, or they can use the full Java language if they want to perform advanced operations in JSP pages.

JavaServer Page Request Model

HTTP requests are processed under the JSP model. In the basic request model, a request is sent directly to a JSP page. Figure 3 illustrates the flow of information in this model. JSP code controls interactions with JavaBeans components for business and data logic processing, and then displays the results in dynamically generated HTML mixed with static HTML code.

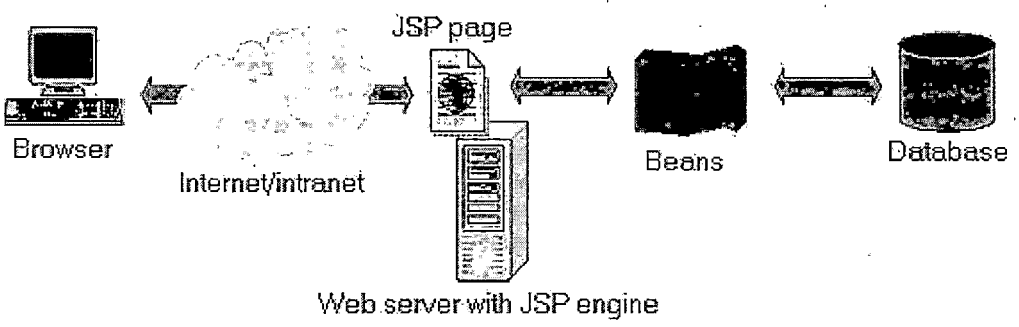


Figure 3. Basic JavaServer Page Request Model

The beans depicted can be JavaBeans or EJB (Enterprise JavaBeans) components. Other, more

complicated request models include calling out to other JSP pages or Java servlets from the requested JSP page.

In this project, I wrote branch.jsp, broker.jsp, brokerprofile.jsp and etc (See Figure 4 and Appendix B) to handle the user's input from the browser and display the dynamic result. For example, in the office search page, when customers select one of the SweetHome's offices, the branch.jsp will handle the request and display the result.

JSP files actually get compiled into Servlets. Some benefits can be acquired by writing JSP instead of just the Servlets. For example, Java and the JSP extensions assist in making the HTML more functional. Servlets on the other hand allow outputting of HTML, but it is a tedious process. The other benefit is easier to make a change in the HTML format.

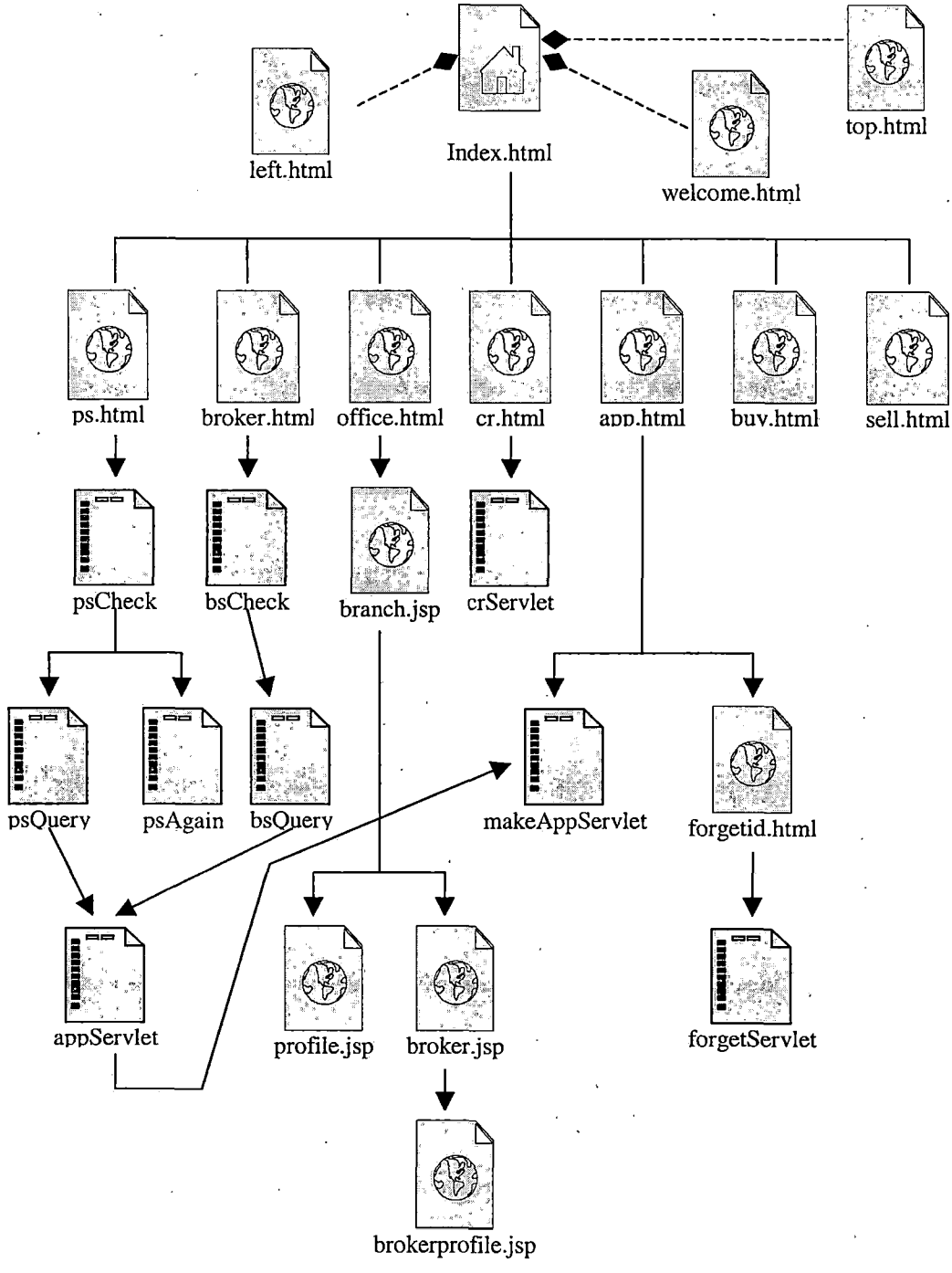


Figure 4. JavaServer Page and Servlet Flow Diagram

3.3 Java Servlet

Servlet are modules of Java code that run in a server application (hence the name "Servlets" similar to "Applets" on the client side) to answer client requests and extend request/response oriented servers, such as Java-enabled web servers. Even tough servlets are not tied to a specific client-server protocol, but they are most commonly used with HTTP and the word "Servlet" is often used in the meaning of "HTTP Servlet".

In this project, appServlet, makeAppservlet, and forgetidServlet etc (See Figure 4 and Appendix C) are responsible for taking data in an HTML order-entry form and applying the business logic used to query or update SweetHome's database. For Example, in the appointment set up page, makeAppServlet provides functions that check the customers enter both their customer and broker Id, and also check whether appointment date is in the past or date is valid, e.g. February 31 is not a valid date.

Since Servlets are written in the highly portable Java language and follow a standard framework, they

provide a means to create sophisticated server extensions in a server and operating system independent way. Typical uses for HTTP Servlets include:

- Processing and/or storing data submitted by an HTML form.
- Providing dynamic content, e.g. returning the results of a database query to the client.
- Managing state information on top of the stateless HTTP, e.g. for an online shopping cart system which manages shopping carts for many concurrent customers and maps every request to the right customer.

Servlets provide a component-based, platform-independent method for building web-based applications, without the performance limitations of other programs. And unlike proprietary server extension mechanisms, Servlets are server and platform independent. This leaves you free to adopt a "best of breed" strategy for your servers, platforms, and tools.

3.4 JavaBean

A JavaBean is a reusable component. Beans are a platform-neutral architecture for the Java application environment. It's the ideal choice for developing or assembling network-aware solutions for different hardware and operating system environments--within the enterprise or across the Internet. In fact, it's the only component architecture you should consider if you're developing for the Java platform.

The JavaBean component architecture extends "Write Once, Run Anywhere" capability to reusable component development. In fact, the JavaBean architecture takes interoperability a major step forward--your code runs on every OS and also within any application environment. A beans developer secures a future in the emerging network software market without losing customers that use proprietary platforms, because JavaBeans components interoperate with ActiveX. JavaBeans architecture connects via bridges into other component models such as ActiveX. Software components that use JavaBeans APIs are thus portable to containers

including Internet Explorer, Visual Basic, Microsoft Word, Lotus Notes, and others.

In this project, I write accessBean, PropertyInfo and BrokerInfo (See Figure 15 and Appendix C) as a reusable component for SweetHome system. For example, every time the psQuery servlet or makeappServlet want to connect to the database, they can just reuse accessBean instead of writing a new one.

JavaBeans is a complete component model. It supports the standard component architecture features of properties, events, methods, and persistence. In addition, JavaBeans provides support for introspection (to allow automatic analysis of a JavaBeans component) and customization (to make it easy to configure a JavaBeans component).

CHAPTER FOUR

SOFTWARE DESIGN

4.1 Use Case and Functions

The following figure shows a Use Case diagram that graphically depicts the users and principal functions of SweetHome. The functions and the actors in the diagram are further described in the next.

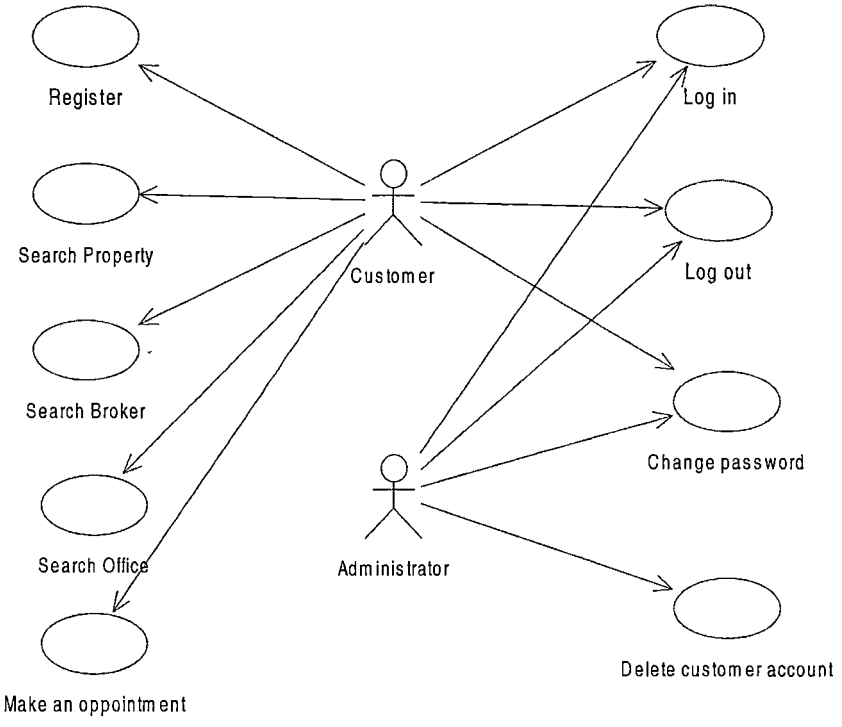


Figure 5. Use Case Diagram

Login and Logout

All users can login to and logout of the SweetHome system. Customers must register first before login to the system.

Change Password

All users are able to change their passwords. The system administrator, on the other hand, can change all users' passwords. This is for the circumstance when a user forgets his password and needs help from the administrator.

Search Property

Customer is asked to enter the search criteria, including the location, property type, number of bedrooms and bathrooms desired, property age, price range, and property size. The system will return the results that meet the searching criteria.

Register

Customers can register their personal information such as SSN, name, address and telephone number into the SweetHome system.

Search Broker

This function will ask the customers to specify their searching criteria that include the broker's

language, experience, sales record, and the location of the broker's office. The system will return the results, and the customer can select a broker with whom to set up an appointment in the appointment set up page.

Search Office

This function will list all the branches of SweetHome. Customers can choose one of them and get the branch's address and telephone number.

Make an Appointment

The customer fills in the broker ID and his customer ID, selects the appointment time, and click the submit button. The system will return either the message "Sorry, the broker is not available at that time" or the message "Congratulations, your appointment has been confirmed."

Delete Customer Account

The administrator can delete customer accounts if necessary.

4.2 Scenarios and User Interface

The following scenario demonstrates a few ways that the SweetHome site could be used by describing a user's view of interaction with the systems.

1. A customer connects to the application by pointing the browser to the URL for the application's home page (See Figure 6).

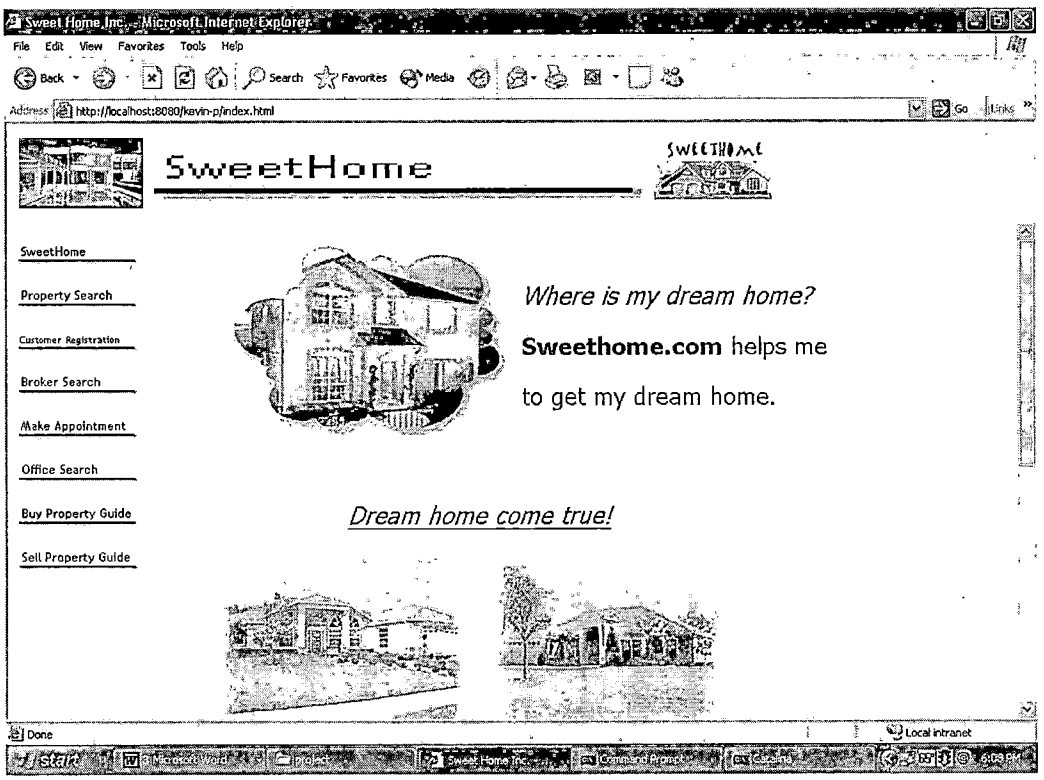


Figure 6. Home Page

This allows the customer to browse through the buyer's guide that will redirect the customer

to some search interface.

2. At any point during the whole interaction, the customers can click the "Customer Registration" to register themselves in the SweetHome web site if they have not registered yet (See Figure 7).

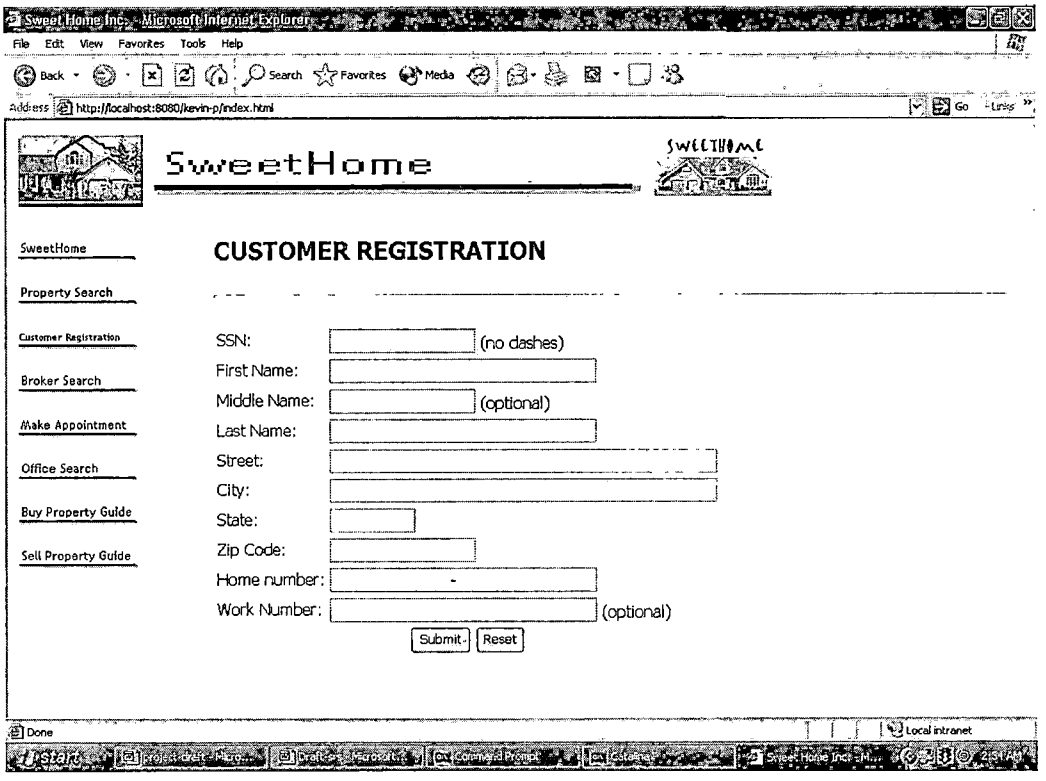


Figure 7. Customer Registration Page

The customer fills in their personal information. If the format is wrong, the application will display a warning page and

require the customer to fill the information in the proper format. If the customer has already registered, the application will remind them. After the registration is completed, the customer can use their SSN as ID to browse the whole site.

3. The customer wanting to search the properties in the SweetHome Inc. can click the "Property Search" bar in the left side. The customer is asked to enter the searching criteria, including the location, property type, number of bedroom required, number of bathroom required, property age, price range, and property size (See Figure 8). The application will return the result that meets the searching criteria. The customer can select one of the properties and click the summit button to make an appointment with the listed broker.

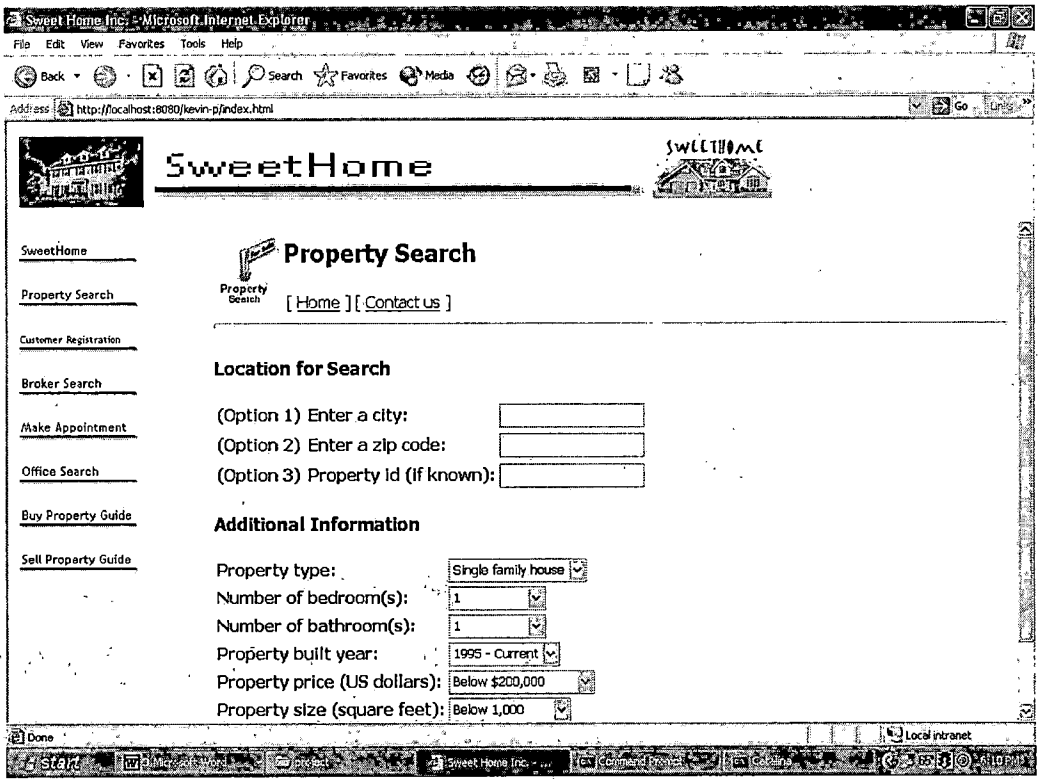


Figure 8. Property Search Page

4. The customer decides to find a broker in SweetHome, Inc. The broker search function will ask the customer to specify their searching criteria that include the language, experience, sale record, and the broker's office (See Figure 9). The application will return the result, and the customer can select one broker with whom to set up an appointment to discuss their needs.

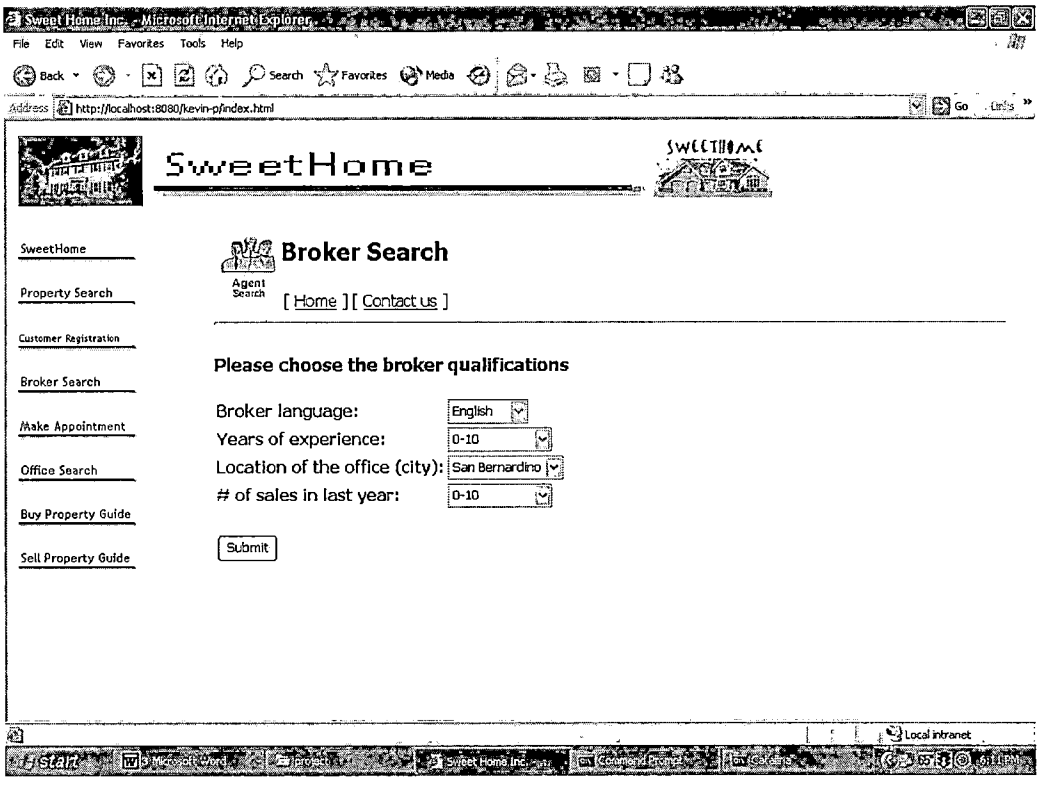


Figure 9. Broker Search Page

5. The customer wants to find branch information in SweetHome (See Figure 10). The customer selects one office and clicks summit button to view the office information like the address, office phone number, fax number and map.

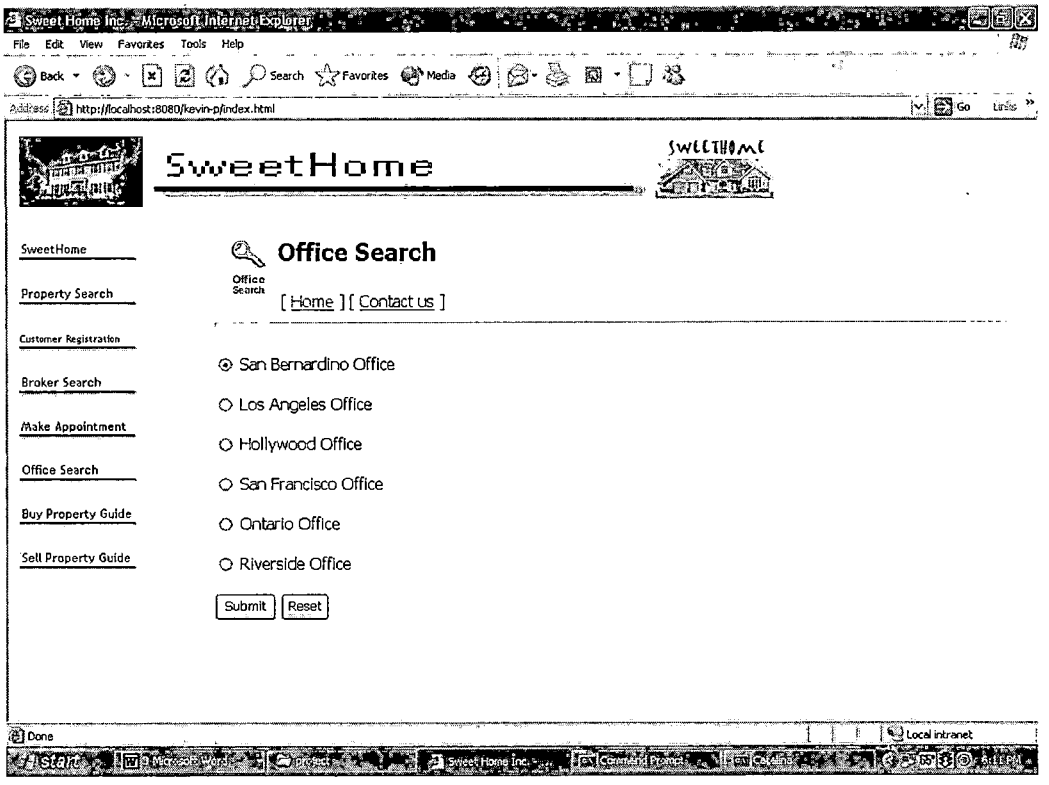


Figure 10. Office Search Page

Another choice from the office information page file is to view the broker profile in that office. If the customer chooses to view the broker profile, the application will return the brokers list in that office. The customer can select one of them to view the broker's profile and then to set up an appointment.

6. At any point during the whole interaction, the customers can click "Make Appointment" to set

up an appointment with the broker (See Figure 11). First, the customer fills in the broker ID and his customer ID, selects the appointment time and clicks the submit button.

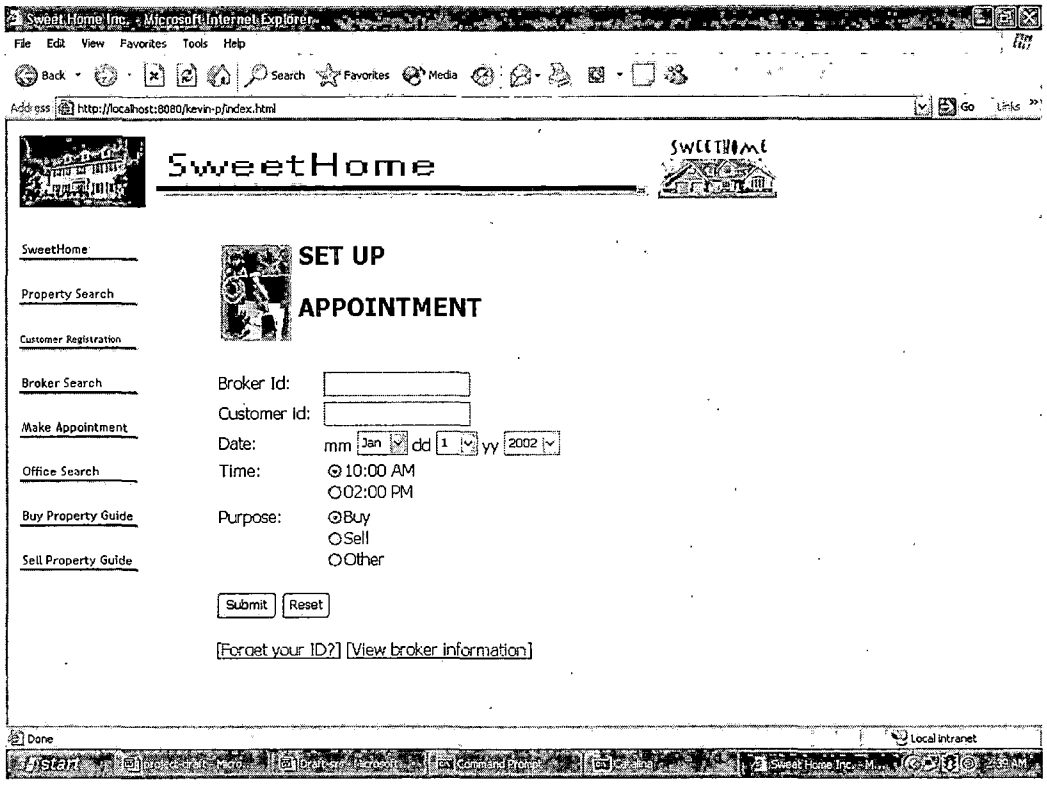


Figure 11. Appointment Set Up Page

The application will return either the message "Sorry, the broker is not available at that time" or the message "Congratulations, your appointment has been confirmed." If the customer chooses a time slot that is in the

past, the message will show "Error! The time you've chosen was in the past. Please choose another time."

The customer clicks the "Forget ID?" link to remember his ID if the customer forgets the customer ID. The customer inputs their SSN and zip code and clicks the submit button. The application will return the condition of the customer and how to use the customerID.

Above is the typical real estate service scenario in SweetHome. If the customers are new to the site, they can start from buy property guide and sell property guide to get more information in SweetHome (See Figure 12 and 13).

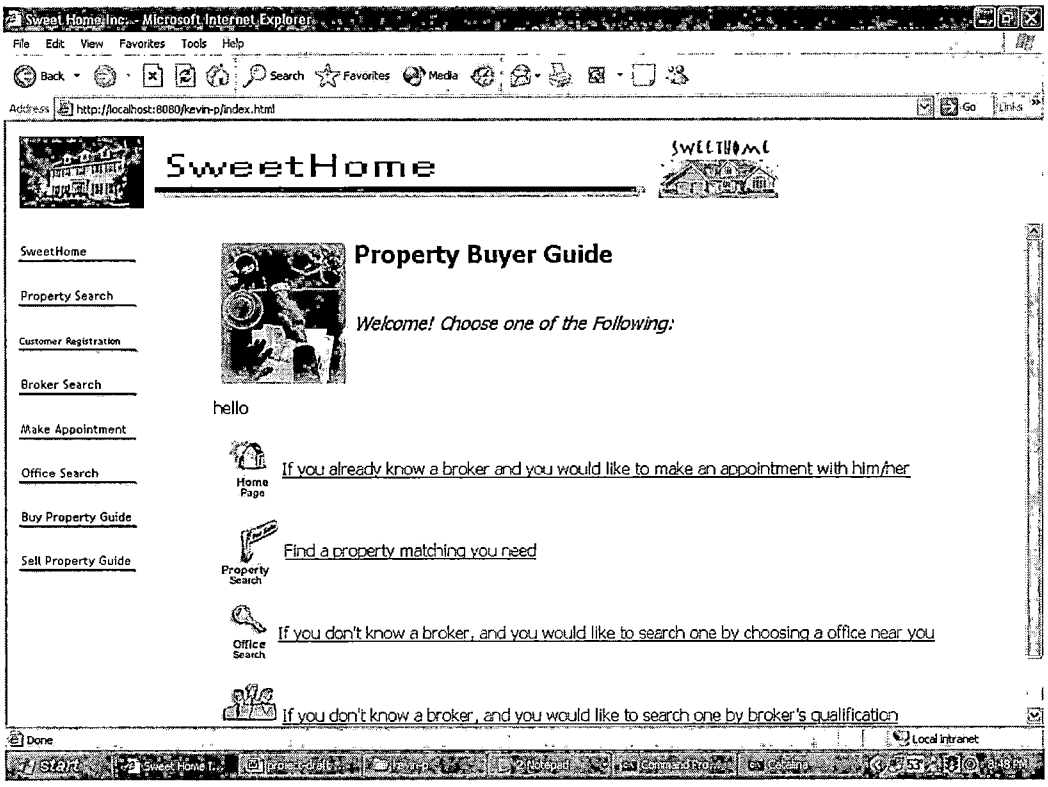


Figure 12. Buyer Guide Page

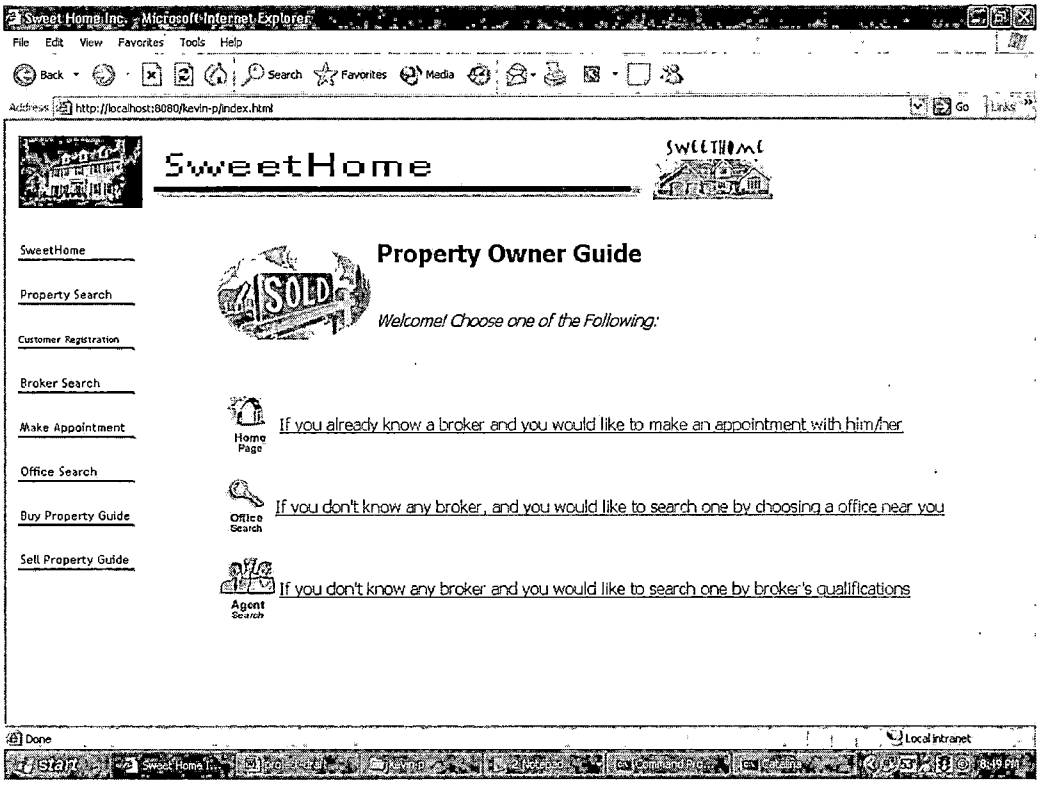


Figure 13. Owner Guide Page

4.3 Deployment Diagram and Class Diagram

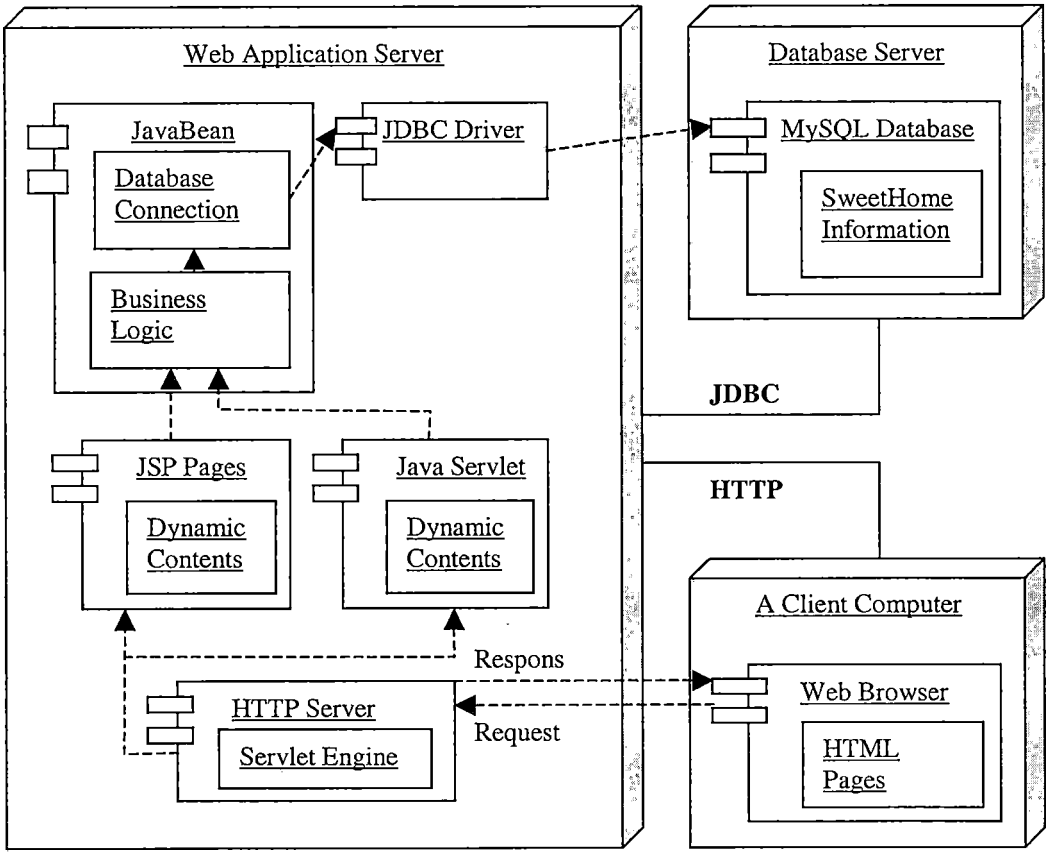


Figure 14. Deployment Diagram

Customer can see HTML page through browser in their computers. The customers' requests are transmitted to the web server via HTTP protocol. The Servlet Engine will load and execute the JSP and JavaServlert to response the customers' requests and display the dynamic content. JavaBean will handle the

business logic, and it can connect to the database by JDBC.

Class Relationship

The figure below briefly shows the relationship of each class in SweetHome system.

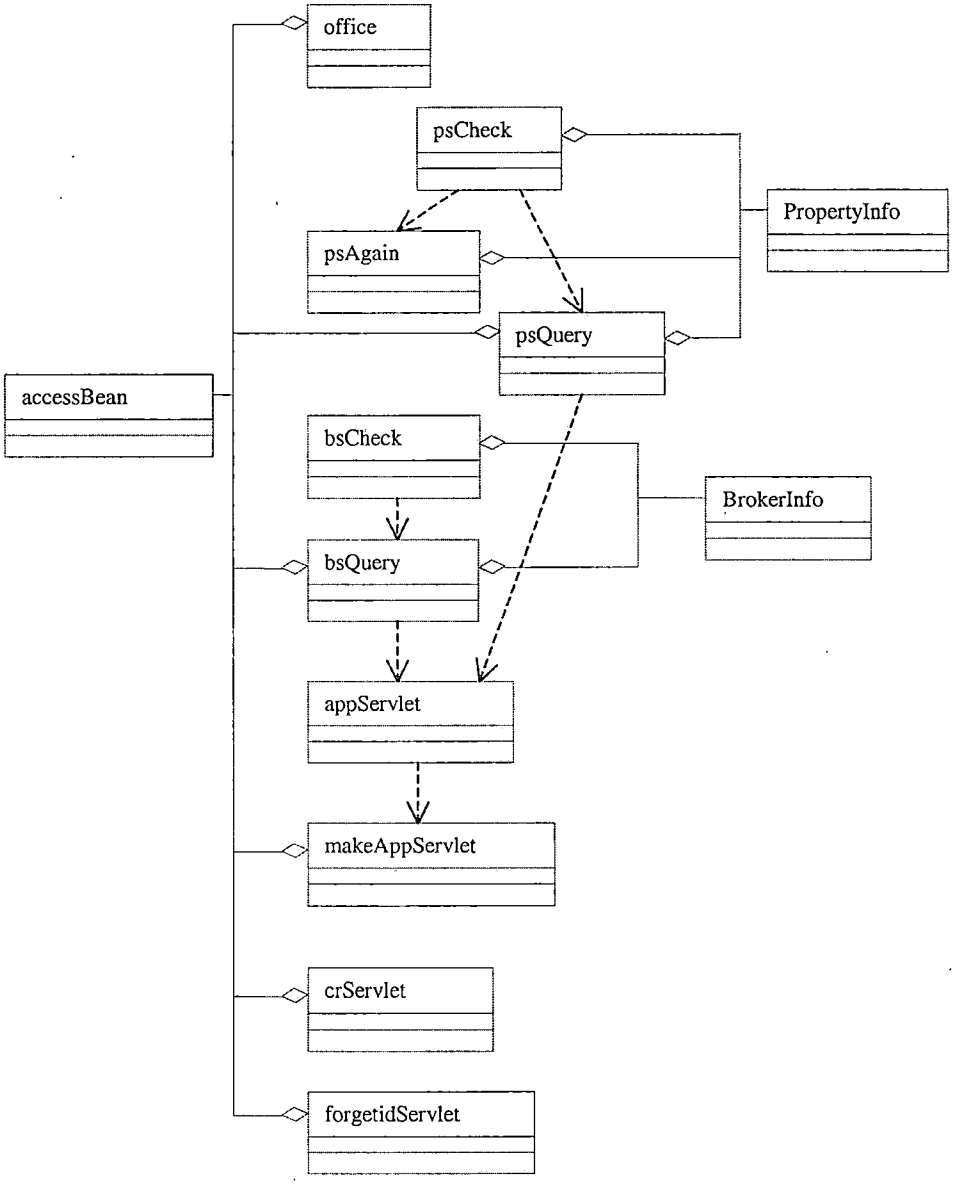


Figure 15. Class Diagram-1

Class Specification

The figures below show each class's detailed attributes and operations. The java source code for each class is in Appendix C.

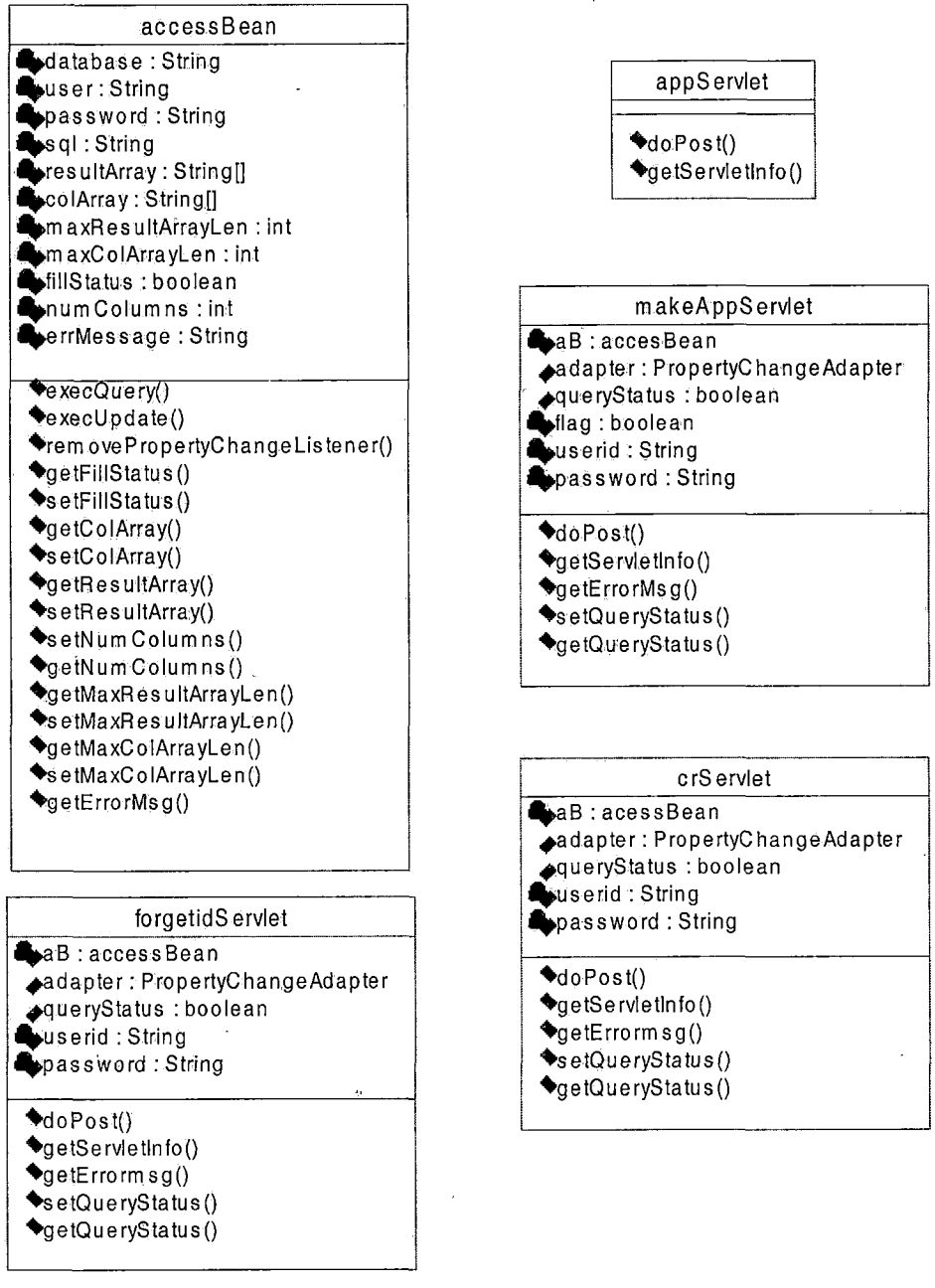


Figure 16. Class Diagram-2

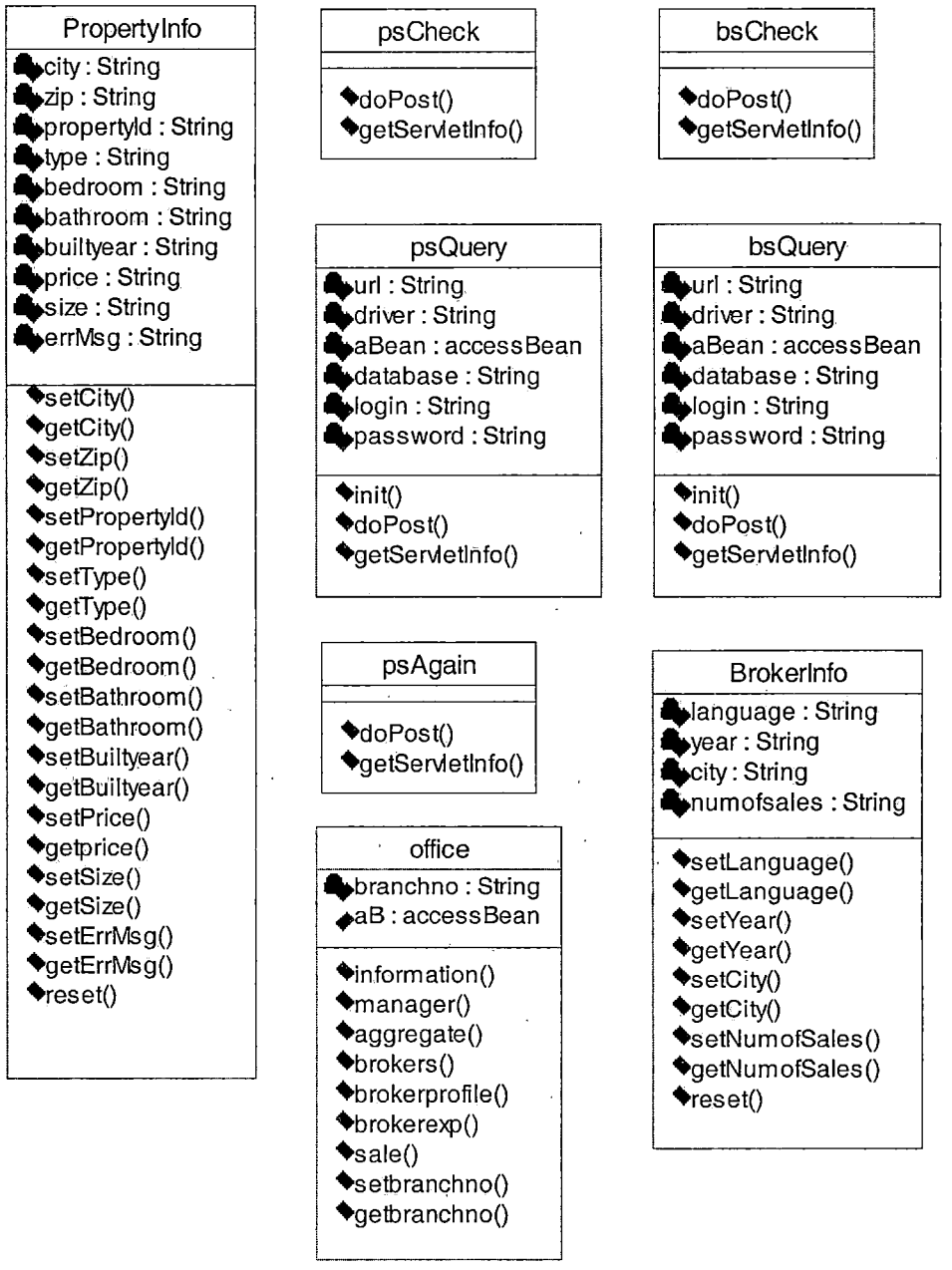


Figure 17. Class Diagram-3

CHAPTER FIVE

DATABASE DESIGN

5.1 Data Requirements

The following figure shows the Data requirements that SweetHome needs.

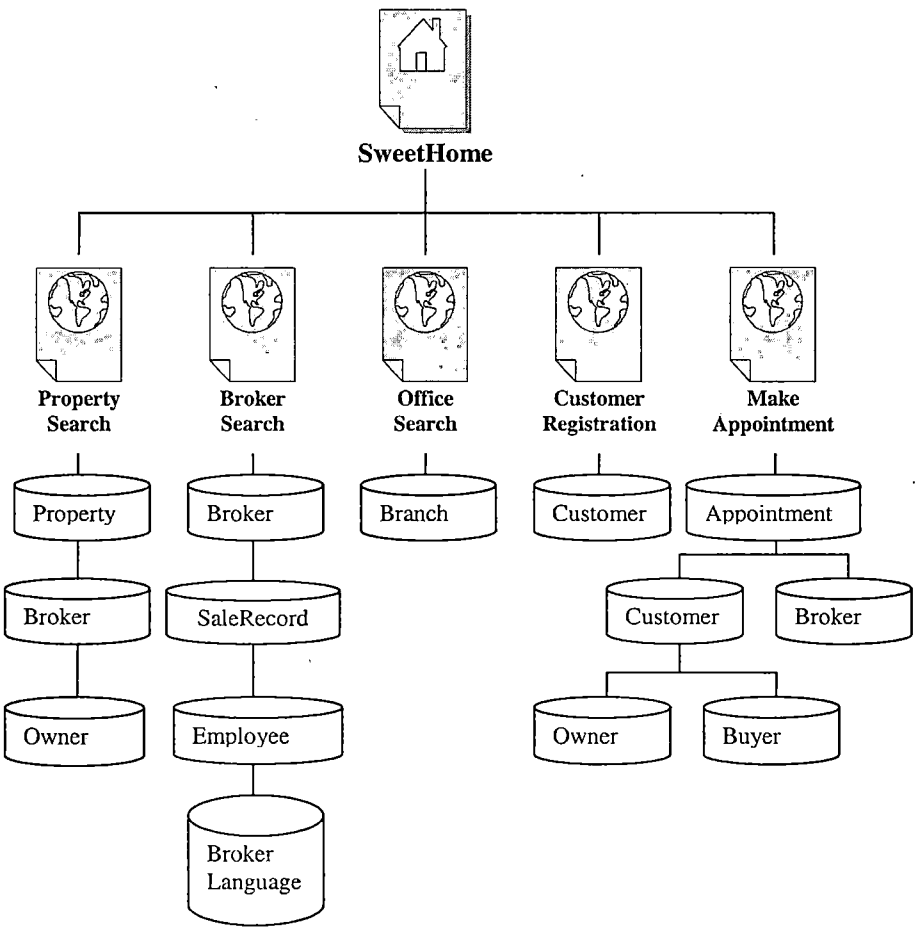


Figure 18. Data Requirement Diagram

Branch Office

Each branch office is identified by a unique branch number and has an address (number, street, city, post code), telephone number and fax number.

Employee

Each branch has employees. Each employee has an SSN, uniquely across all branch offices. Information held on each employee includes name (first name, middle name, and last name), sex, salary, address (number, street, city, post code), SSN, telephone number, date of birth, job title, languages they can speak, and the starting date they were employed.

Brokers have information on the broker license number, pager number, and the starting date they were employed. Each broker has unique broker ID across branches.

Property

Each property has property ID, property type (ex: condo, town house, and single family house), sale price, square feet, number of rooms, number of baths, built year and address (number, street, city, post code). The years of the property can be calculated

from the built year. Property ID can uniquely identify the property.

Customer

Customers who want to buy or sell the house have to register with SweetHome and make appointments with SweetHome brokers. Each customer is allowed to make appointments with only one broker in SweetHome at a time. Each customer who registers with SweetHome has SSN, phone number (home number and work number), address (number, street, city, post code), name (first name, middle name, and last name) and registered date with the branch. SSN can uniquely identify customers. Customers in SweetHome database are all property buyers and property owners. Customers that currently have properties to sell or that would like to buy the houses are considered as active customers in the database.

Appointment

Property owners provide property information to brokers. Brokers will arrange an appointment with property buyers on available properties. Appointment dates, time, and comments will be recorded with every

appointment. Appointment date and time uniquely identifies appointments between one customer and one broker.

SaleRecord

The branch keeps all transactions in the records. Records have information on transaction number, property number; broker's SSN who made the deal, transaction amount, and the date. The transaction number can uniquely identify records within a branch.

5.2 Database Relational Model

Table 3. Property Table

PropertyId	Price	SquareFt	BuiltYear	Type

NoRoom	NoBath	Street	City	State

Code

MySQL results

Query conditions were met by 87 rows.

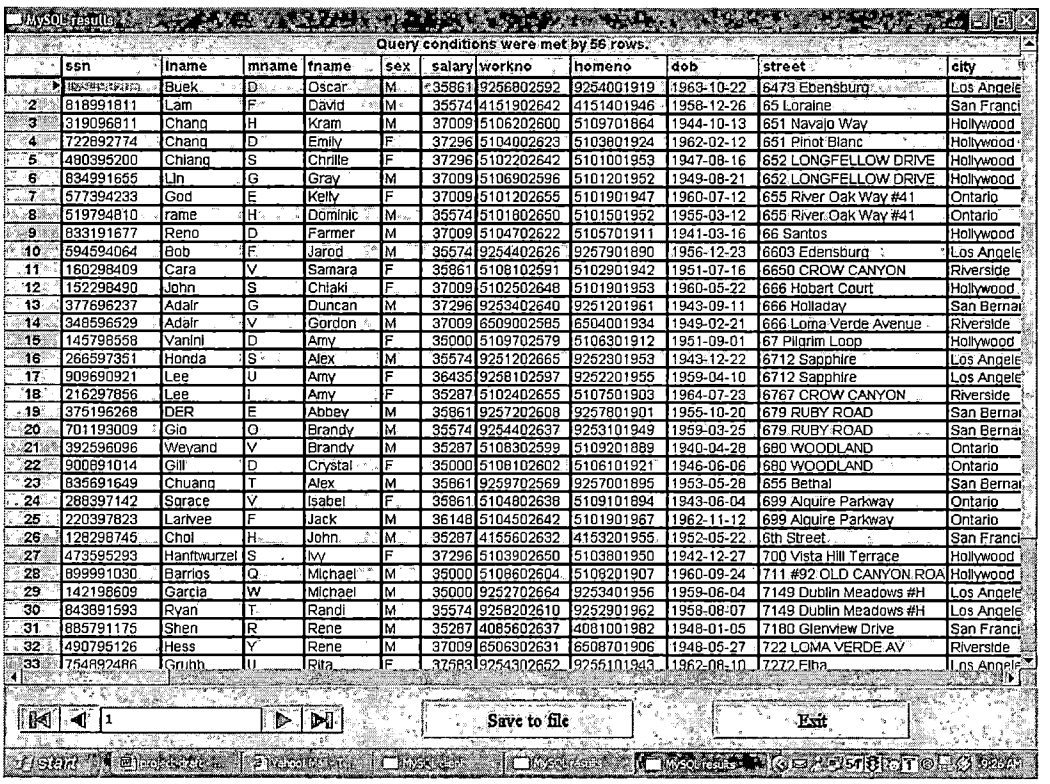
	propertyid	price	squareft	builtyear	type	noroom	nobath	street	city	state	co
▶	[illegible]	200000	1500	1992	Single Family House	5	2	100 Kendall Dr	San Bernardino	CA	92
2	00002	300000	2500	1996	Single Family House	8	3	100 College Ave	San Bernardino	CA	92
3	04701	353008	1040	1963	Condo	3	2	3140 MIDDLEFIELD AVENUE	Los Angeles	CA	96
4	58401	227440	1110	1946	Condo	3	2	3146 Brent Street	Riverside	CA	94
5	57328	390940	1020	1970	Condo	2	1	3148 MARKWOOD COURT	Hollywood	CA	95
6	64432	383800	1270	1963	Condo	2	1	36 MARKWOOD COURT	Hollywood	CA	95
7	54404	538684	1660	1986	Single family house	4	3	3159 Huntington Street	Riverside	CA	94
8	70453	262876	1160	1959	Condo	3	2	85 Huntington Street	Riverside	CA	94
9	74628	315112	1380	1974	Condo	2	1	3161 Sydney Way	Riverside	CA	94
10	62910	294832	1020	1940	Condo	2	1	3170 GREENTREE WY	San Bernardino	CA	95
11	15169	418852	1730	1975	Town house	2	1	3173 Orwell Place	Los Angeles	CA	96
12	59677	400876	1750	1972	Town house	4	3	3179 Terry Street	Riverside	CA	94
13	77646	492568	1410	1982	Town house	4	3	3183 Bruce Drive	Los Angeles	CA	96
14	68713	224896	1020	1949	Condo	2	1	3204 MATTOS AV	San Bernardino	CA	95
15	64998	330664	1360	1975	Condo	2	1	3208 Hancock Street	Los Angeles	CA	96
16	50814	518356	1790	1994	Single family house	4	3	3229 Keith Avenue	Riverside	CA	94
17	75652	381952	1200	1977	Condo	3	2	323 James St.	Hollywood	CA	92
18	50296	438688	1550	1974	Town house	2	1	32399 Lake Temescal	Los Angeles	CA	96
19	08587	444568	1700	1982	Town house	2	1	3246 Magdalena Place	Riverside	CA	94
20	67644	528064	2000	1985	Single family house	2	1	32481 Salton Sea	Los Angeles	CA	96
21	65691	348724	1200	1960	Single family house	3	2	65 Jelencic Drive	Ontario	CA	95
22	67311	234016	1000	1943	Condo	3	2	3250 Jelencic Drive	Ontario	CA	95
23	65025	310960	1180	1969	Single family house	2	1	3255 Elm	Hollywood	CA	92
24	55165	541396	1870	1980	Single family house	2	1	7 Ocie Street	Ontario	CA	95
25	53064	508852	1920	1987	Single family house	4	3	327 Ocie Street	Ontario	CA	95
26	61715	403288	1510	1976	Town house	4	3	3274 SAN RIVAS DR	San Bernardino	CA	95
27	22754	424612	1580	1974	Town house	4	3	328 Leland Avenue	San Bernardino	CA	95
28	63909	474112	1720	1973	Town house	3	2	3280 Wyndham Drive	Los Angeles	CA	96
29	71808	560428	1930	1989	Single family house	4	3	32842 SHAVER LAKE	Los Angeles	CA	96
30	74858	453616	1510	1983	Town house	4	3	330 Escobar Street	Los Angeles	CA	96
31	60030	565684	1770	1988	Single family house	2	1	330 Sequim Com.	Los Angeles	CA	96
32	73819	334432	1210	1979	Single family house	2	1	3322 KIMBER CT	San Bernardino	CA	95
33	74103	442288	1770	1977	Town house	3	2	3326 Kestrel Place	Los Angeles	CA	96

1 Save to file Exit

Figure 19. Property Data in MySQL

Table 5. Employee Table

SSN	LName	MName	FName	Sex
Salary	WorkNo	HomeNo	DOB	Street
City	State	Code	StartDate	BranchNo



	ssn	lname	mname	fname	sex	salary	workno	homeno	dob	street	city
1	[illegible]	Buek	D	Oscar	M	35861	9256802592	9254001919	1963-10-22	6473 Ebensburg	Los Angele
2	818991811	Lam	F	David	M	35574	4151902642	4151401946	1958-12-26	65 Loraine	San Franci
3	319096811	Chang	H	Kram	M	37009	5106202600	5109701864	1944-10-13	651 Navajo Way	Hollywood
4	722892774	Chang	D	Emily	F	37296	5104002623	5103801924	1962-02-12	651 Pinot Blanc	Hollywood
5	490395200	Chiang	S	Chrille	F	37296	5102202642	5101001953	1947-08-16	652 LONGFELLOW DRIVE	Hollywood
6	834991655	Lin	G	Gray	M	37009	5106902596	5101201952	1949-08-21	652 LONGFELLOW DRIVE	Hollywood
7	577394233	God	E	Kelly	F	37009	5101202655	5101901947	1960-07-12	655 River Oak Way #41	Ontario
8	519794810	rame	H	Dominic	M	35574	5101802650	5101501952	1955-03-12	655 River Oak Way #41	Ontario
9	833191677	Reno	D	Farmer	M	37009	5104702622	5105701911	1941-03-16	66 Santos	Hollywood
10	594594064	Bob	F	Jarod	M	36574	9254402626	9257901890	1956-12-23	6603 Edensburg	Los Angele
11	160298409	Cara	V	Samara	F	35861	5108102591	5102901942	1951-07-16	6650 CROW CANYON	Riverside
12	152298490	John	S	Chiaki	F	37009	5102502648	5101901953	1960-05-22	666 Hobart Court	Hollywood
13	377696237	Adair	G	Duncan	M	37296	9253402640	9251201961	1943-09-11	666 Holladay	San Bernar
14	348596529	Adair	V	Gordon	M	37009	6509002565	6504001934	1949-02-21	666 Loma Verde Avenue	Riverside
15	145798558	Vanini	D	Amy	F	36000	5109702579	5106301912	1951-09-01	67 Pilgrim Loop	Hollywood
16	266597351	Honda	S	Alex	M	35574	9251202665	9252301953	1943-12-22	6712 Sapphire	Los Angele
17	909690921	Lee	U	Amy	F	36436	9258102597	9252201955	1959-04-10	6712 Sapphire	Los Angele
18	216297856	Lee	I	Amy	F	35287	5102402655	5107501903	1964-07-23	6767 CROW CANYON	Riverside
19	376196268	DER	E	Abbey	M	35861	9257202608	9257801901	1955-10-20	679 RUBY ROAD	San Bernar
20	701193009	Gio	O	Brandy	M	35574	9254402637	9253101949	1959-03-25	679 RUBY ROAD	San Bernar
21	392596096	Weyand	V	Brandy	M	35287	5108302599	5109201889	1940-04-28	680 WOODLAND	Ontario
22	900891014	Gill	D	Crystal	F	35000	5108102602	5106101921	1946-06-06	680 WOODLAND	Ontario
23	836691649	Chuang	T	Alex	M	35861	9259702569	9257001895	1953-05-28	655 Bethal	San Bernar
24	288397142	Sgrace	V	Isabel	F	35861	5104802638	5109101894	1943-06-04	699 Alquire Parkway	Ontario
25	220397823	Larivee	F	Jack	M	36148	5104502642	5101901967	1962-11-12	699 Alquire Parkway	Ontario
26	128298745	Choi	H	John	M	35287	4155602632	4153201955	1952-05-22	6th Street	San Franci
27	473595293	Hanftwurzel	S	Ivy	F	37296	5103902650	5103801950	1942-12-27	700 Vista Hill Terrace	Hollywood
28	899991030	Barrios	Q	Michael	M	35000	5108602604	5108201907	1960-09-24	711 #92 OLD CANYON ROA	Hollywood
29	142198609	Garcia	W	Michael	M	35000	9252702664	9253401956	1959-06-04	7149 Dublin Meadows #H	Los Angele
30	843891593	Ryan	T	Randi	M	35574	9258202610	9252901962	1958-08-07	7149 Dublin Meadows #H	Los Angele
31	885791175	Shen	R	Rene	M	35267	4085602637	4081001982	1948-01-05	7180 Glenview Drive	San Franci
32	490795126	Hess	Y	Rene	M	37009	6506302631	6508701906	1948-05-27	722 LOMA VERDE AV	Riverside
33	754892486	Gruhn	U	Rita	F	37683	9254302652	9255101943	1962-08-10	7272 Elba	Los Angele

Figure 21. Employee Data in MySQL

Table 6. Broker Table

SSN	BrokerId	BLicenseNo	PageNo	DateExpStart

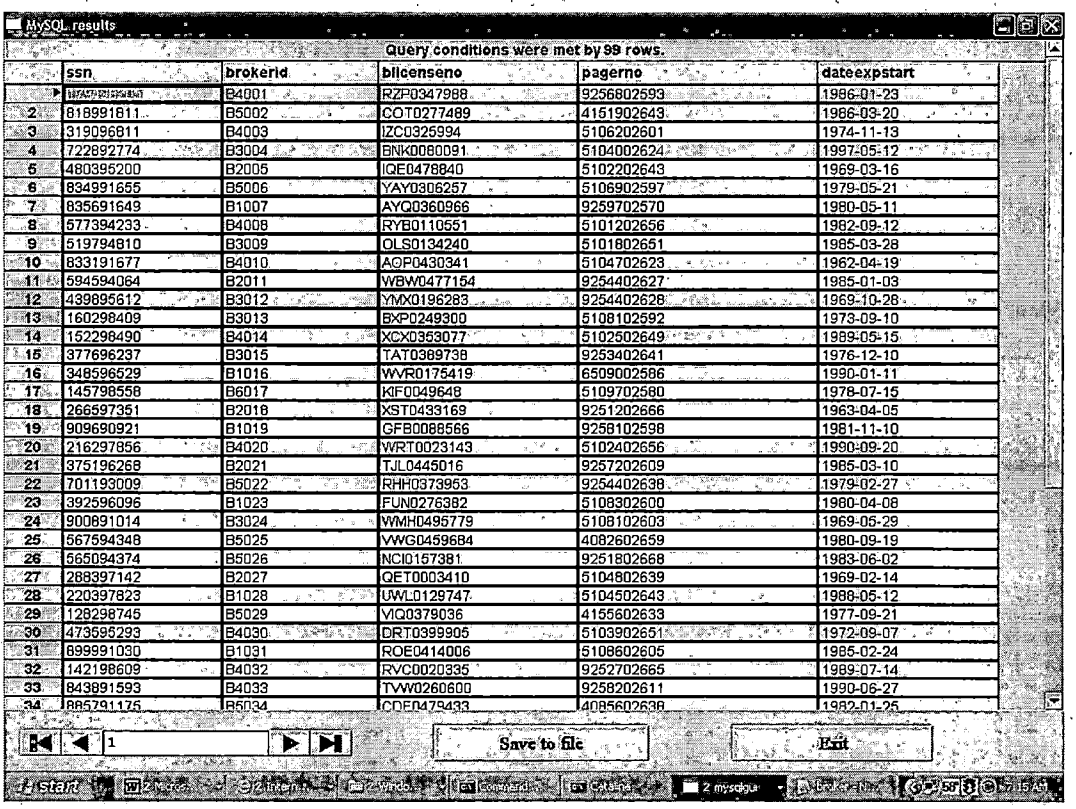


Figure 22. Broker Data in MySQL

Table 7. BrokerLanguage Table

BrokerId	Language

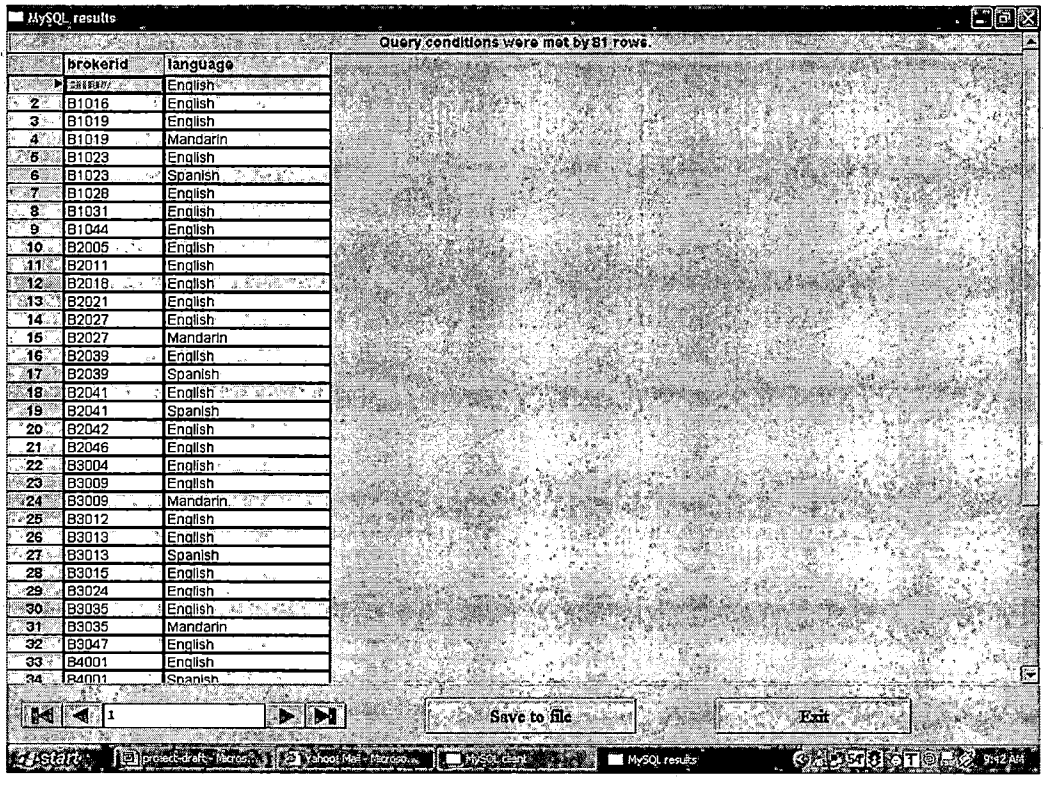
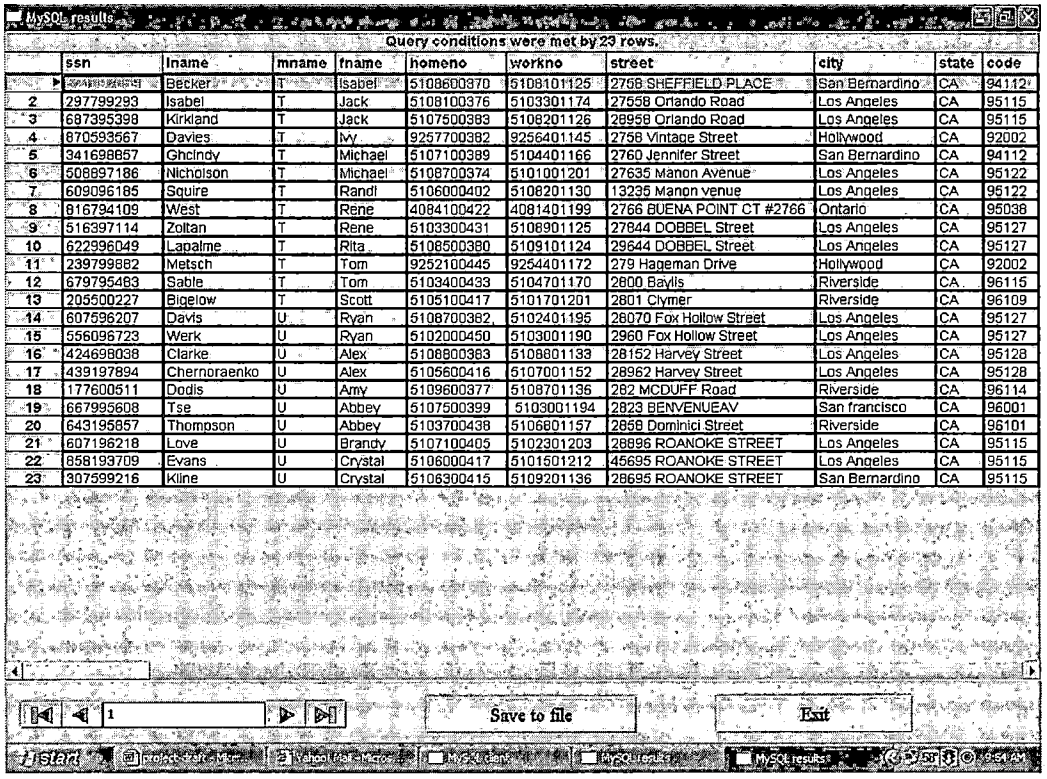


Figure 23. BrokerLanguage Data in MySQL

Table 8. Customer Table

SSN	LName	MName	FName	HomeNo
WorkNo	Street	City	State	Code
RegDate				



	ssn	lname	mname	fname	homeno	workno	street	city	state	code
1	[illegible]	Becker	T	Isabel	5108600370	5108101125	2758 SHEFFIELD PLACE	San Bernardino	CA	94112
2	297799293	Isabel	T	Jack	5108100376	5103301174	27658 Orlando Road	Los Angeles	CA	95115
3	687395398	Kirkland	T	Jack	5107500383	5108201126	28958 Orlando Road	Los Angeles	CA	95115
4	870593567	Davies	T	Ivy	9257700382	9256401145	2758 Vintage Street	Hollywood	CA	92002
5	341698857	Ghcindy	T	Michael	5107100389	5104401166	2760 Jennifer Street	San Bernardino	CA	94112
6	508897186	Nicholson	T	Michael	5108700374	5101001201	27635 Manon Avenue	Los Angeles	CA	95122
7	609096185	Squire	T	Randl	5106000402	5108201130	13235 Manon venue	Los Angeles	CA	95122
8	816794109	West	T	Rene	4084100422	4081401199	2766 BUENA POINT CT #2766	Ontario	CA	95038
9	516397114	Zoltan	T	Rene	5103300431	5108901125	27844 DOBBEL Street	Los Angeles	CA	95127
10	622996049	Lapalme	T	Rita	5108500380	5109101124	29644 DOBBEL Street	Los Angeles	CA	95127
11	239799882	Metsch	T	Tom	9252100445	9254401172	279 Hageman Drive	Hollywood	CA	92002
12	679795483	Sable	T	Tom	5103400433	5104701170	2800 Baylis	Riverside	CA	96115
13	205500227	Bigelow	T	Scott	5105100417	5101701201	2801 Clymer	Riverside	CA	96109
14	607596207	Davis	U	Ryan	5108700382	5102401195	26070 Fox Hollow Street	Los Angeles	CA	95127
15	556096723	Werk	U	Ryan	5102000450	5103001190	2960 Fox Hollow Street	Los Angeles	CA	95127
16	424698038	Clarke	U	Alex	5108800383	5108801133	28152 Harvey Street	Los Angeles	CA	95128
17	439197894	Chernoraenko	U	Alex	5105600416	5107001152	28962 Harvey Street	Los Angeles	CA	95128
18	177600511	Dodis	U	Amy	5109600377	5108701136	262 MCDUFF Road	Riverside	CA	96114
19	667995608	Tse	U	Abbey	5107500399	5103001194	2823 BENVENUEAV	San francisco	CA	96001
20	643195857	Thompson	U	Abbey	5103700438	5106801157	2858 Dominici Street	Riverside	CA	96101
21	607196218	Love	U	Brandy	5107100405	5102301203	28896 ROANOKE STREET	Los Angeles	CA	95115
22	858193709	Evans	U	Crystal	5106000417	5101501212	45695 ROANOKE STREET	Los Angeles	CA	95115
23	307599216	Kline	U	Crystal	5106300415	5109201136	28695 ROANOKE STREET	San Bernardino	CA	95115

Figure 24. Customer Data in MySQL

Table 9. Appointment Table

BrokerId	CustomerSSN	AppDate	AppTime	AppComment

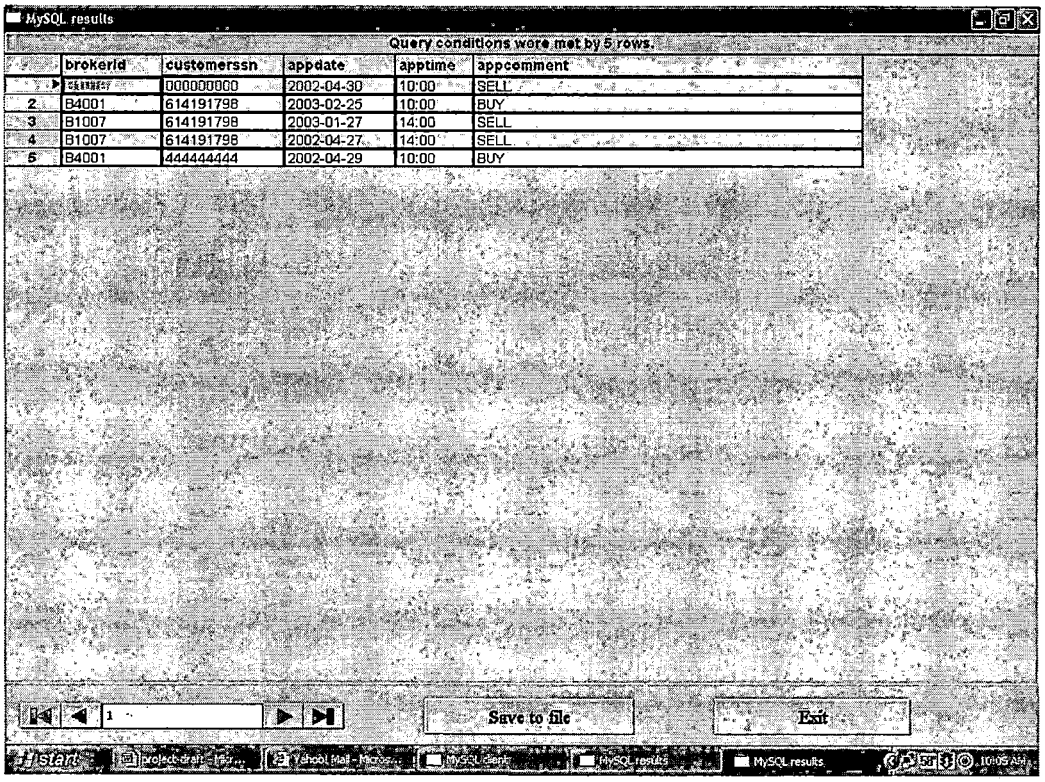


Figure 25. Appointment Data in MySQL

Table 10. Buyer Table

SSN	Active

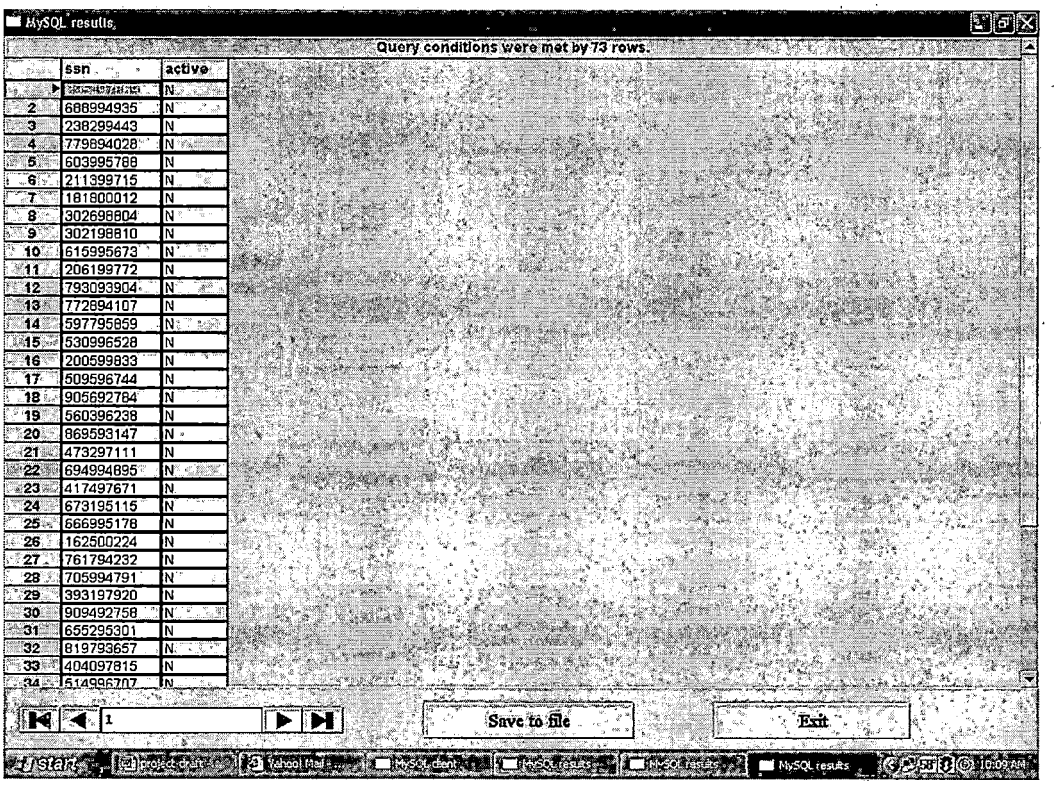


Figure 26. Buyer Data in MySQL

Table 11. Owner Table

SSN	Active	BrokerId

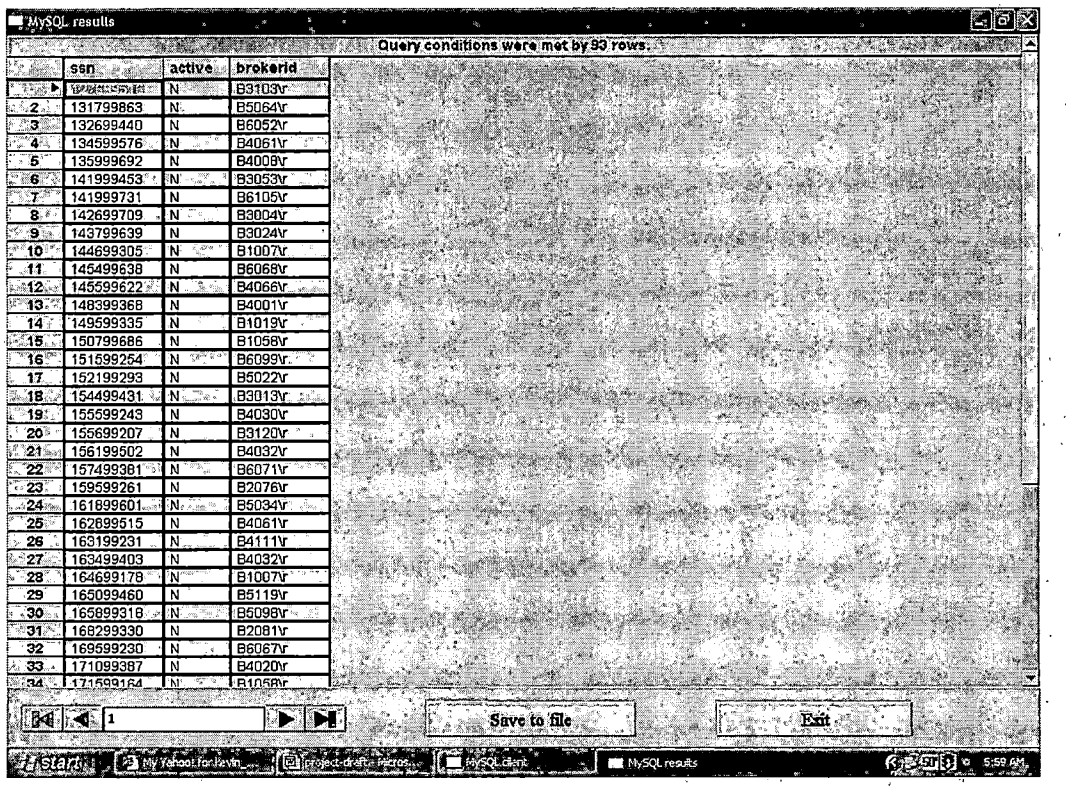
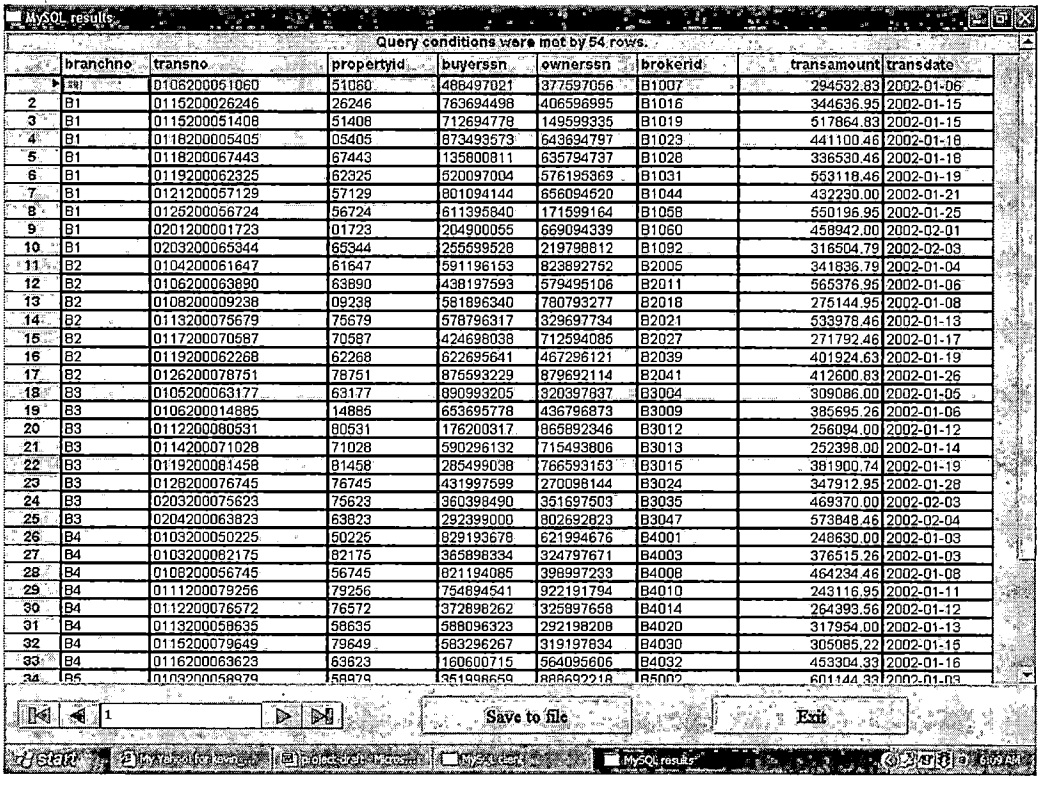


Figure 27. Owner Data in MySQL

Table 12. SaleRecord Table

BranchNo	TransNo	PropertyId	BuyerSSN	OwnerSSN
BrokerId	TransAmount	TransDate		



	branchno	transno	propertyid	buyerssn	ownerssn	brokerid	transamount	transdate
▶	B1	0106200051060	51060	486497021	377597056	B1007	294532.83	2002-01-06
2	B1	0115200026246	26246	763694498	406596995	B1016	344636.95	2002-01-15
3	B1	0115200051408	51408	712694778	149599335	B1019	517864.83	2002-01-15
4	B1	0118200005405	05405	673493573	643694797	B1023	441100.46	2002-01-18
5	B1	0118200067443	67443	135800811	635794737	B1026	336530.46	2002-01-18
6	B1	0119200062325	62325	520097004	576195369	B1031	553118.46	2002-01-19
7	B1	0121200057129	57129	801094144	656094520	B1044	432230.00	2002-01-21
8	B1	0125200056724	56724	611395840	171599164	B1058	550196.95	2002-01-25
9	B1	0201200001723	01723	204900055	669094339	B1060	458942.00	2002-02-01
10	B1	0203200065344	65344	255699528	219798812	B1092	316504.79	2002-02-03
11	B2	0104200061647	61647	591196153	823892752	B2005	341836.79	2002-01-04
12	B2	0106200063890	63890	438197593	579495106	B2011	565376.95	2002-01-06
13	B2	0108200009238	09238	581896340	780793277	B2018	275144.95	2002-01-08
14	B2	0113200075679	75679	578796317	329697734	B2021	533978.46	2002-01-13
15	B2	0117200070587	70587	424698038	712594086	B2027	271792.46	2002-01-17
16	B2	0119200062268	62268	622695641	467296121	B2039	401924.63	2002-01-19
17	B2	0126200078751	78751	875593229	879692114	B2041	412600.63	2002-01-26
18	B3	0105200063177	63177	890993205	320397837	B3004	309086.00	2002-01-05
19	B3	0106200014885	14885	653695778	436796873	B3009	385695.26	2002-01-06
20	B3	0112200080531	80531	176200317	865892346	B3012	256094.00	2002-01-12
21	B3	0114200071028	71028	590296132	715493806	B3013	252398.00	2002-01-14
22	B3	0119200081458	81458	285499038	766593153	B3015	381900.74	2002-01-19
23	B3	0128200076745	76745	431997599	270098144	B3024	347912.95	2002-01-28
24	B3	0203200075623	75623	360398490	351697503	B3035	469370.00	2002-02-03
25	B3	0204200063823	63823	292399000	802692823	B3047	573848.46	2002-02-04
26	B4	0103200050225	50225	829193678	621994676	B4001	248630.00	2002-01-03
27	B4	0103200082175	82175	365898334	324797671	B4003	376515.26	2002-01-03
28	B4	0108200056745	56745	821194085	398997233	B4008	464234.46	2002-01-08
29	B4	0111200079256	79256	754894541	922191794	B4010	243116.95	2002-01-11
30	B4	0112200076572	76572	372698262	325897658	B4014	264393.56	2002-01-12
31	B4	0113200058635	58635	588096323	292198208	B4020	317954.00	2002-01-13
32	B4	0115200079649	79649	583296267	319197834	B4030	305085.22	2002-01-15
33	B4	0116200063623	63623	160600715	564095606	B4032	453304.33	2002-01-16
34	B5	0103200058979	58979	351998659	888692218	B5002	601144.33	2002-01-03

Figure 28. SaleRecord Data in MySQL

Structured Query Language for Branch Table

```
CREATE TABLE BRANCH(
BRANCHNO  VARCHAR(4)     NOT NULL,
TELNO     VARCHAR(20)    UNIQUE NOT NULL,
FAXNO     VARCHAR(20),
STREET    VARCHAR(30)    NOT NULL,
CITY      VARCHAR(20)    NOT NULL,
STATE     VARCHAR(2)     NOT NULL,
CODE      VARCHAR(5)     NOT NULL,
LICENSENO VARCHAR(15),
CONSTRAINT BRANCHNO_BRANCH_PK PRIMARY KEY(BRANCHNO),
CONSTRAINT ADDRESS_BRANCH_U UNIQUE(STREET, CITY,
STATE, CODE));
```

Structured Query Language for Employee Table

```
CREATE TABLE EMPLOYEE(
SSN       VARCHAR(9)     NOT NULL,
LNAME     VARCHAR(20)    NOT NULL,
MNAME     VARCHAR(20),
FNAME     VARCHAR(20)    NOT NULL,
SEX       CHAR           NOT NULL,
```

```
SALARY      INT(15)          NOT NULL,
WORKNO      VARCHAR(20)      NOT NULL,
HOMENO      VARCHAR(20)      NOT NULL,
DOB         DATE,
STREET      VARCHAR(30)      NOT NULL,
CITY        VARCHAR(20)      NOT NULL,
STATE       VARCHAR(2)       NOT NULL,
CODE        VARCHAR(5)       NOT NULL,
STARTDATE   DATE             NOT NULL,
BRANCHNO    VARCHAR(4)       NOT NULL,
CONSTRAINT SSN_EMP_PK PRIMARY KEY(SSN),
CONSTRAINT SEX_EMP_CK CHECK(SEX IN('F','M')),
CONSTRAINT BRANCHNO_EMP_FK FOREIGN KEY(BRANCHNO)
REFERENCES BRANCH(BRANCHNO) ON DELETE CASCADE);
```

Structured Query Language for
Broker Table

```
CREATE TABLE BROKER(
SSN             VARCHAR(9)      NOT NULL,
BROKERID        VARCHAR(9)      UNIQUE NOT NULL,
BLICENSENO      VARCHAR(15)     UNIQUE NOT NULL,
PAGERNO         VARCHAR(20)     UNIQUE NOT NULL,
DATEEXPSTART    DATE            NOT NULL,
CONSTRAINT SSN_BROKER_PK PRIMARY KEY(SSN),
```

```
CONSTRAINT SSN_BROKER_FK FOREIGN KEY(SSN) REFERENCES
EMPLOYEE(SSN)  ON DELETE CASCADE);
```

Structured Query Language for Brokerlanguage Table

```
CREATE TABLE BROKERLANGUAGE(
BROKERID VARCHAR(9)  NOT NULL,
LANGUAGE VARCHAR(15) NOT NULL,
CONSTRAINT BROKERID_LANG_PK PRIMARY KEY(BROKERID,
LANGUAGE),
CONSTRAINT BROKERID_LANG_FK FOREIGN KEY(BROKERID)
REFERENCES BROKER(BROKERID) ON DELETE CASCADE);
```

Structured Query Language for Property Table

```
CREATE TABLE PROPERTY(
PROPERTYID      VARCHAR(10)     NOT NULL,
PRICE           INT(8)          NOT NULL,
SQUAREFT        INT(5)          NOT NULL,
BUILTYEAR       INT(4)          NOT NULL,
TYPE            VARCHAR(20)     NOT NULL,
NOROOM          INT(2)          NOT NULL,
NOBATH          INT(2)          NOT NULL,
STREET          VARCHAR(30)     NOT NULL,
CITY            VARCHAR(20)     NOT NULL,
```

```
STATE           VARCHAR(2)      NOT NULL,
CODE            VARCHAR(5)      NOT NULL,
CONSTRAINT PROPERTYID_PRO_PK PRIMARY KEY(PROPERTYID),
CONSTRAINT SQUAREFT_PRO_CK CHECK(SQUAREFT>0),
CONSTRAINT BUILTYEAR_PRO_CK CHECK(BUILTYEAR>1900),
CONSTRAINT ADDRESS_PRO_U UNIQUE(STREET, CITY, STATE,
CODE));
```

Structured Query Language for Customer Table

```
CREATE TABLE CUSTOMER(
SSN         VARCHAR(9)      NOT NULL,
LNAME       VARCHAR(20)     NOT NULL,
MNAME       VARCHAR(20),
FNAME       VARCHAR(20)     NOT NULL,
HOMENO      VARCHAR(20)     NOT NULL,
WORKNO      VARCHAR(20),
STREET      VARCHAR(30)     NOT NULL,
CITY        VARCHAR(20)     NOT NULL,
STATE       VARCHAR(20)     NOT NULL,
CODE        VARCHAR(5)      NOT NULL,
REGDATE     DATE            NOT NULL,
CONSTRAINT SSN_CUSTOMER_PK PRIMARY KEY(SSN));
```

Structured Query Language for
Buyer Table

```
CREATE TABLE BUYER(

SSN     VARCHAR(9) NOT NULL,

ACTIVE CHAR(1) NOT NULL,

CONSTRAINT SSN_BUYER_PK PRIMARY KEY(SSN),

CONSTRAINT SSN_BUYER_FK FOREIGN KEY(SSN) REFERENCES

CUSTOMER(SSN) ON DELETE CASCADE,

CONSTRAINT ACTIVE_BUYER_CK CHECK(ACTIVE IN('Y','N')));
```

Structured Query Language for
Owner Table

```
CREATE TABLE OWNER(

SSN         VARCHAR(9)      NOT NULL,

ACTIVE      CHAR(1)         NOT NULL,

BROKERID    VARCHAR(9)      NOT NULL,

CONSTRAINT SSN_OWNER_PK PRIMARY KEY(SSN),

CONSTRAINT SSN_OWNER_FK FOREIGN KEY(SSN) REFERENCES

CUSTOMER(SSN) ON DELETE CASCADE,

CONSTRAINT BID_OWNER_FK FOREIGN KEY(BROKERID)

REFERENCES BROKER(BROKERID),

CONSTRAINT ACTIVE_OWNER_CK CHECK(ACTIVE IN('Y','N')));
```

Structured Query Language for Appointment Table

```
CREATE TABLE APPOINTMENT(
BROKERID        VARCHAR(9)      NOT NULL,
CUSTOMERSSN     VARCHAR(9)      NOT NULL,
APPDATE         DATE            NOT NULL,
APPTIME         VARCHAR(5)      NOT NULL,
APPCOMMENT      VARCHAR(100)    NOT NULL,
CONSTRAINT APPOINTMENT_PK PRIMARY KEY(BROKERID,
APPDATE, APPTIME),
CONSTRAINT BROKERID_APP_FK FOREIGN KEY(BROKERID)
REFERENCES BROKER(BROKERID) ON DELETE CASCADE,
CONSTRAINT CUSTIMERSSN_APP_FK FOREIGN KEY(CUSTOMERSSN)
REFERENCES CUSTOMER(SSN) ON DELETE CASCADE);
```

Structured Query Language for Salerecord Table

```
CREATE TABLE SALERECORD(
BRANCHNO        VARCHAR(4)      NOT NULL,
TRANSNO         VARCHAR(16)     NOT NULL,
PROPERTYID      VARCHAR(10)     NOT NULL        UNIQUE,
BUYERSSN        VARCHAR(9)      NOT NULL,
OWNERSSN        VARCHAR(9)      NOT NULL,
BROKERID        VARCHAR(9)      NOT NULL,
```

```
TRANSAMOUNT     INT(11)          NOT NULL,
TRANSDATE       DATE             NOT NULL,
CONSTRAINT BNO_TNO_SR_PK PRIMARY KEY(BRANCHNO,
TRANSNO),
CONSTRAINT BNO_SR_FK FOREIGN KEY(BRANCHNO) REFERENCES
BRANCH(BRANCHNO) ON DELETE CASCADE,
CONSTRAINT BID_SR_FK FOREIGN KEY(BROKERID) REFERENCES
BROKER(BROKERID) ON DELETE CASCADE,
CONSTRAINT PID_SR_FK FOREIGN KEY(PROPERTYID)
REFERENCES PROPERTY(PROPERTYID) ON DELETE CASCADE,
CONSTRAINT OWNERSSN_SR_FK FOREIGN KEY(OWNERSSN)
REFERENCES OWNER(SSN) ON DELETE CASCADE,
CONSTRAINT BUYERSSN_SR_FK FOREIGN KEY(BUYERSSN)
REFERENCES BUYER(SSN) ON DELETE CASCADE);
```

5.4 Entity Relationship Diagram

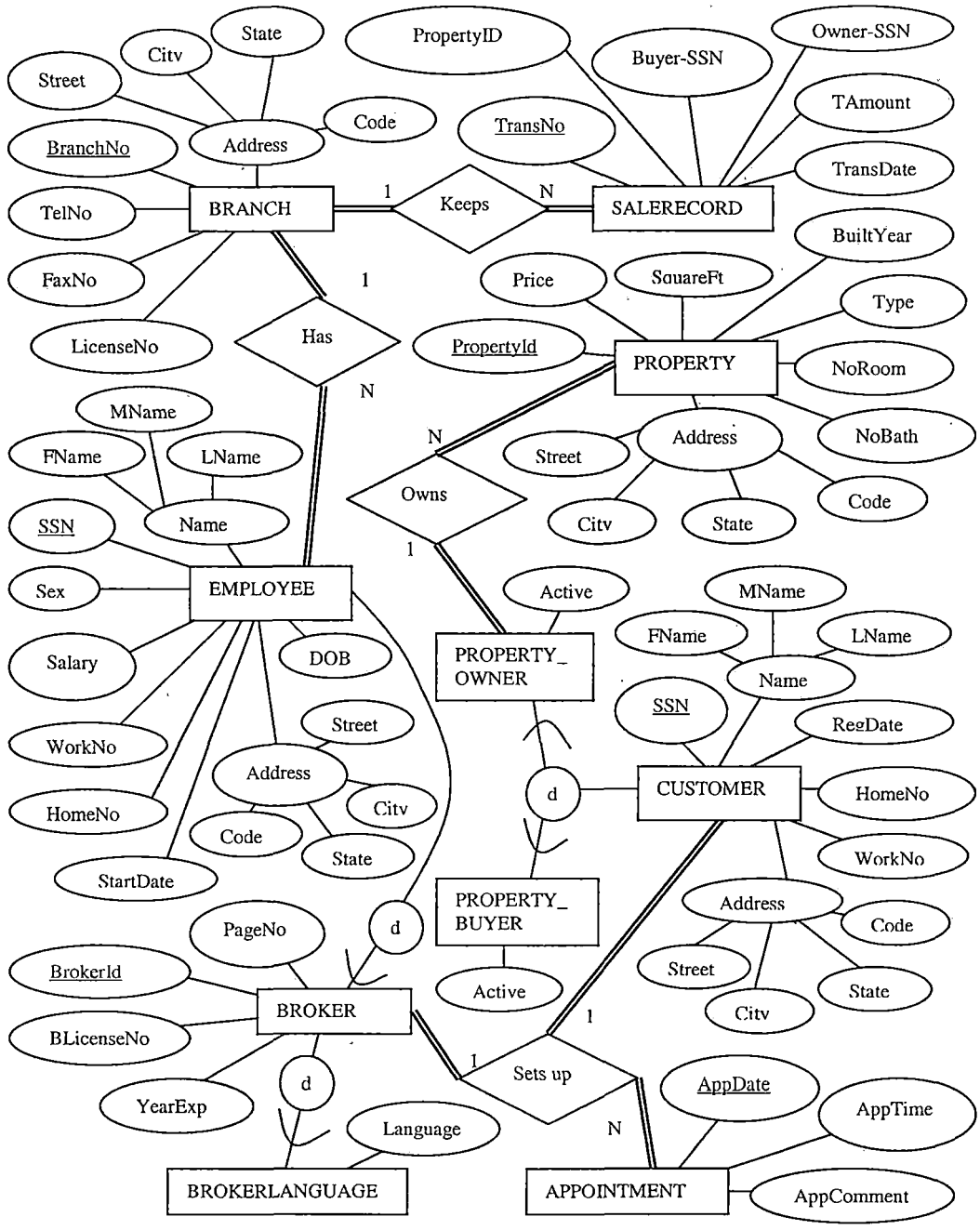


Figure 29. Entity Relationship Diagram

CHAPTER SIX

SOFTWARE TEST AND SYSTEM MAINTENCE

6.1 Software Test

Table 13. Test Table 1

Testing Content: Home Page				
No	Testing Case	Expect Result	Verified	Comment
1	Click SweetHome	Go to SweetHome home page	√	
2	Click Property Search	Go to property search page	√	
3	Click Customer Registration	Go to customer registration page	√	
4	Click Broker Search	Go to broker search page	√	
5	Click Make Appointment	Go to appointment set up page	√	
6	Click Office Search	Go to office search page	√	
7	Click Buy Property Guide	Go to buyer guide page	√	
8	Click Buy Property Guide	Go to owner guide page	√	

Table 14. Test Table 2

Testing Content: Property Search Page				
No	Testing Case	Expect Result	Verified	Comment
1	City: Hollywood	Successful Query	√	
2	Zip: 92002	Successful Query	√	
3	Property ID: 57328	Successful Query	√	
4	City: Riverside Type: Condo	Successful Query	√	
5	Zip: 94120 Type: Single family house Bedrooms:4	Successful Query	√	
6	Property id: 67311 Bathroom:2 Year: Before 1970 Price:$200,000 -$299,999	Successful Query	√	
7	City:San Francisco Type: Condo Bathroom: 1 Year: Before 1970 size: 1,000-1,499	Successful Query	√	
8	Zip: 96116 Type: Condo Bedroom: 3 Bathroom: 2 Year: Before 1970 size: 1,000-1,499	Successful Query	√	

9	Lack City, Zip and Property id	Error Message	√	Need to specify a location for search or a valid property id

Table 15. Test Table 3

Testing Content: Customer Registration Page				
No	Testing Case	Expect Result	Verified	Comment
1	Lack SSN	Error Message	√	Input column can't be empty
2	Lack First Name	Error Message	√	Input column can't be empty
3	Lack Middle Name	Registration Success	√	
4	Lack Last Name	Error Message	√	Input column can't be empty

5	Lack Street	Error Message	√	Input column can't be empty
6	Lack City	Error Message	√	Input column can't be empty
7	Lack State	Error Message	√	Input column can't be empty
8	Lack Code	Error Message	√	Input column can't be empty
9	Lack Home Number	Error Message	√	Input column can't be empty
10	Lack Work Number	Registration Success	√	

Table 16. Test Table 4

Testing Content: Broker Search Page				
No	Testing Case	Expect Result	Verified	Comment
1	language: English experience: 0-10 location: San Bernardino sales: 11 and above	Successful Query	√	
2	language: mandarin experience: 11-20 location: Hollywood sales: 11-20	Successful Query	√	
3	language: Spanish experience: 21 and above location: Ontario sales: 11 and above	Successful Query	√	
4	language: Mandarin experience: 11-20 location: Riverside sales: 0-10	Successful Query	√	

Table 17. Test Table 5

Testing Content: Office Search Page				
No	Testing Case	Expect Result	Verified	Comment
1	San Bernardino Office	San Bernardino Office Information	√	
2	Los Angeles Office	Los Angeles Office Information	√	
3	Hollywood Office	Hollywood Office Information	√	
4	San Francisco Office	San Francisco Office Information	√	
5	Ontario Office	Ontario Office Information	√	
6	Riverside Office	Riverside Office Information	√	

Table 18. Test Table 6

Testing Content: Appointment Set Up Page (Test Date: May/1)				
No	Testing Case	Expect Result	Verified	Comment
1	Lack Broker ID	Error Message	√	Broker ID is Required
2	Lack Customer ID	Error Message	√	Customer ID is Required
3	Date: Feb/29/2002	Error Message	√	Invalid Date
4	Date: Feb/30/2003	Error Message	√	Invalid Date
5	Date: Feb/31/2002	Error Message	√	Invalid Date
6	Date: Apr/31/2003	Error Message	√	Invalid Date
7	Date: Jun/31/2002	Error Message	√	Invalid Date
8	Date: Sep/31/2003	Error Message	√	Invalid Date
9	Date: Non/31/2002	Error Message	√	Invalid Date
10	Date: Jan/10/2002	Error Message	√	Date is in the past
11	Date: May/1/2002	Error Message	√	Can not make appointment today
12	Date: Jan/10/2003	Successful Message	√	

Table 19. Test Table 7

Testing Content: Buy Property Guide Page				
No	Testing Case	Expect Result	Verified	Comment
1	Click first choice	Go to appointment set up page	√	
2	Click Second choice	Go to property search page	√	
3	Click third choice	Go to office search page	√	
4	Click fourth choice	Go to broker search page	√	

Table 20. Test Table 8

Testing Content: Sell Property Guide Page				
No	Testing Case	Expect Result	Verified	Comment
1	Click first choice	Go to appointment set up page	√	
2	Click second choice	Go to office search page	√	
3	Click third choice	Go to broker search page	√	

6.2 System Maintenance

Operating System: MS Windows 98/NT/2000/XP or Red Hat Linux

Browser: Netscape Navigator 3.0 or higher and Microsoft's Internet Explorer 3.x or higher.

Memory: 128 MB recommended

Free disk space: 500 MB recommended

Development Tools: Java Development Kit (j2sdk1.4.0)

(Download from http://java.sun.com/j2se/1.4/download.html)

Web Server: Jakarta-tomcat-4.0.1

(Download from http://jakarta.apache.org/builds/jakarta-tomcat-4.0/release/v4.0.1/src/)

Database: MySQL (mysql-3.23.47-win.zip)

(Download from http://www.mysql.com/downloads/mysql-3.23.html)

JDBC: mm.mysql.jdbc-1.2c.tar

(Download from http://www.mysql.com/downloads/api-jdbc.html)

Mysqlgui: mysqlgui-win32-static-1.7.5-2

(Download from http://www.mysql.com/downloads/gui-mysqlgui.html)

SweetHome's HTML File: See Appendix A

SweetHome's JSP File: See Appendix B

SweetHome's JavaServlet and JavaBean File: See Appendix C

Table 21. Source Code Table

HTML File	JSP File
index.html left.html top.html welcome.html ps.html broker.html office.html cr.html app.html buy.html sell.html forgetid.html	branch.jsp broker.jsp profile.jsp brokerprofile.jsp
JavaServlet File	**JavaBean File**
appServlet.java makeAppServlet.java crservlet.java forgetidServlet.java psCheck.java psQuery.java psAgain.java bsCheck.java bsQuery.java	accessBean.java PropertyInfo.java BrokerInfo.java office.java

CHAPTER SEVEN

FUTURE DIRECTIONS

The future directions for the SweetHome website are as follows:

1. Use J2EE to develop higher security and more reliability on EJB reusable component.
2. Add more functions to the SweetHome website, such as adding a little calendar on the Appointment set up page or sending a notification e-mail when customers make an appointment successfully, and so on.
3. Use a more powerful database such as Oracle 9i.
4. Purchase firewall software to protect the system.

To apply SweetHome in the real world, it needs a well organized team and enough expense. This team should include real estate experts, web designers, database administrators, and professional programmers. It also needs to improve performance and security by purchasing a domain name, advanced firewall software, and reliable server hardware.

APPENDIX A

SOURCE CODE: HTML PART

Index.html

```
<html>
<head><title>Sweet Home Inc.</title></head>

 <frameset rows="101,*" framespacing="0" border="0" frameborder="0">
 <frame name="top" scrolling="no" noresize target="contents" src="top.html">
 <frameset cols="195,*">
 <frame name="contents" target="main" src="left.html">
 <frame name="main" src="welcome.html">
 </frameset>
 <noframes>
 <body>
 <p>This page uses frames.</p>
 </body>
 </noframes>
 </frameset>
</html>
```

left.html

```
<html>
<head><title>Left</title><base target="main"></head>
<body>

<p><a href="welcome.html"><img border="0" src="images/sweethome.gif" width="120"
height="25"></a></p>
<p><a href="ps.html"><img border="0" src="images/property.gif" width="120"
height="25"></a></p>
<p><a href="cr.html"><img border="0" src="images/customer.gif" width="120"
height="25"></a></p>
<p><a href="broker.html"><img border="0" src="images/broker.gif" width="120"
height="25"></a></p>
<p><a href="app.html"><img border="0" src="images/appt.gif" width="120"
height="25"></a></p>
<p><a href="office.html"><img border="0" src="images/office.gif" width="120"
height="25"></a></p>
<p><a href="buy.html"><img border="0" src="images/buy.gif" width="120"
height="25"></a></p>
<p><a href="sell.html"><img border="0" src="images/sell.gif" width="120"
height="25"></a></p>
</body>
</html>
```

top.html

```
<html>
<head><title>TOP</title><base target="contents"></head>
```

```
<body>

<p><IMG SRC="images/houses_animation.gif" ALIGN=left><IMG
SRC="images/sweethome.gif" width="500" height="62" ALIGN=center>
<IMG SRC="images/logo_5.gif" ALIGN=center></p>
</body>
</html>
```

welcome.html

```
<html>
<head><title>Sweethome.com Homepage</title></head>
<body>
<p>
<IMG SRC="images/dream-bubble.jpg" WIDTH="300" HEIGHT="205" ALIGN="Left"> <br>
<br>
<p>
<big><big><font><i>Where is my dream home?</i></font></big></big>
<p>
<big><big><font><b></b></font></big></big>
<p>
<big><big><font><b>Sweethome.com</b></font>
</big></big><font><big><big>helps me </big></big></font>
<p>
<font><big><big>to get my dream home.</big></big></font><br>
<br><br><br><br><p>
<table cellpadding="5">
 <caption>
  <font><big><big><ins><i>Dream home come true!</i></ins></big></big></font>
 </caption>
 <tr>
  <td><IMG SRC="images/house1.jpg"></td>
  <td><IMG SRC="images/house2.jpg"></td>
 </tr>
 <tr>
  <td><IMG SRC="images/house4.jpg"></td>
  <td><IMG SRC="images/house3.jpg"></td>
 </tr>
 <tr>
  <td><IMG SRC="images/house5.jpg"></td>
  <td><IMG SRC="images/house6.jpg"></td>
 </tr>
</table>
<p>
<IMG SRC="images/list_link.gif" WIDTH="600" HEIGHT="68" BORDER="0"
ALIGN="Middle">
</body></html>
```

cr.html

```
<html>
<title>CR</title>
<body>
<H2><B>CUSTOMER REGISTRATION</B></H2>
<hr size=5>
<form  method="post" action="./servlet/crServlet">
 <table align=left >
<tr><td>SSN:</td>
<td><input type="text" name="SSN1" size="3"> -
   <input type="text" name="SSN2" size="2"> -
   <input type="text" name="SSN3" size="4"> </td></tr>

 <tr><td>First Name:</td><td><input type=text size=40 name=FIRST ></td></tr>
 <tr><td>Middle Name:</td><td><input type=text size=20 name=MID> (optional)</td></tr>
 <tr><td>Last Name:</td><td><input type=text size=40 name=LAST></td></tr>
 <tr><td>Street:</td><td><input type=text size=60 name=STREET></td></tr>
 <tr><td>City:</td><td><input type=text size=60 name=CITY></td></tr>
 <tr><td>State:</td>
        <td><select name=STATE>
                                        <option value="" selected>
                                    <option value=AK >Alaska
                                    <option value=AL >Alabama
                                    <option value=AR >Arkansas
                                    <option value=AZ >Arizona
                                    <option value=CA >California
                                    <option value=CO >Colorado
                                    <option value=CT >Connecticut
                                    <option value=DE >Delaware
                                    <option value=FL >Florida
                                    <option value=GA >Georgia
                                    <option value=HI >Hawaii
                                    <option value=IA >Iowa
                                    <option value=ID >Idaho
                                    <option value=IL >Illinois
                                    <option value=IN >Indiana
                                   <option value=KS >Kansas
                                   <option value=KY >Kentucky
                                   <option value=LA >Louisiana
                                   <option value=MA >Massachusetts
                                   <option value=MD >Maryland
                                   <option value=ME >Maine
                                   <option value=MI >Michigan
                                   <option value=MN >Minnesota
                                   <option value=MO >Missouri
                                   <option value=MS >Mississippi
                                   <option value=MT >Montana
                                   <option value=NC >North Carolina
                                   <option value=ND >North Dakota
                                   <option value=NE >Nebraska
                                   <option value=NH >New Hampshire
```

```
                                    <option value=NJ >New Jersey
                                    <option value=NM >New Mexico
                                    <option value=NV >Nevada
                                    <option value=NY >New York
                                    <option value=OH >Ohio
                                    <option value=OK >Oklahoma
                                    <option value=OR >Oregon
                                    <option value=PA >Pennsylvania
                                    <option value=RI >Rhode Island
                                    <option value=SC >South Carolina
                                    <option value=SD >South Dakota
                                    <option value=TN >Tennessee
                                    <option value=TX >Texas
                                    <option value=UT >Utah
                                    <option value=VA >Virginia
                                    <option value=VT >Vermont
                                    <option value=WA >Washington
                                    <option value=WI >Wisconsin
                                    <option value=WV >West Virginia
                                    <option value=WY >Wyoming

                    </select>
                </td></tr>

 <tr><td>Zip Code:</td><td><input type=text size=20 name=ZIP></td></tr>

<tr><td>Home Number:</td>
<td><input type="text" name="HOME1" size="3"> -
   <input type="text" name="HOME2" size="3"> -
   <input type="text" name="HOME3" size="4"></td></tr>

<tr><td>Work Number:</td>
<td><input type="text" name="WORK1" size="3"> -
   <input type="text" name="WORK2" size="3"> -
   <input type="text" name="WORK3" size="4"> (optional)</td></tr>

  <tr align=center><td colspan=2>
   <input type=submit name=SEND value=Submit>
   <input type=reset value=Reset></td></tr>
</table>
</form>
</body>
</html>
```

ps.html

```
<html>
<head><title>Property Search</title></head>
<body>

<form method="post" action="./servlet/psCheck">
```

```
<P><IMG SRC="images/property_searchB.gif" ALIGN=left><h2>Property Search</h2></P>
<table >
        <tr>
        <td>[ <a href="welcome.html">Home</a> ]</td>
        <td>[ <a href="mailto:webmaster@sweethome.com">Contact us</a> ]</td>
        </tr>
</table>
<hr size=5>
<br>
<h3>Location for Search</h3>
<table>
<tr>
        <th align=left>(Option 1)</th>
        <th></th>
        <th align=left>Enter a city:</th>
        <td><input type="text" name="city" size="20"></td>
</tr><tr>
        <th align=left>(Option 2)</th>
        <th></th>
        <th align=left>Enter a zip code:</th>
        <td><input type="text" name="zip" size="20"></td>
</tr><tr>
        <th align=left>(Option 3)</th>
        <th></th>
        <th align=left>Property id (if known):</th>
        <td><input type="text" name="id" size="20"></td>
</tr>
</table>
<br>
<h3>Additional Information</h3>
<table>
<tr>
        <th align=left>Property type:</th>
        <td><select size="1.5" name="house">
                <option selected>None</option>
                <option>Single family house</option>
                <option>Town house</option>
                <option>Condo</option>
                </select></td>
</tr><tr>
        <th align=left>Number of bedroom(s):</th>
        <td><select size="1.5" name="bedroom">
                <option selected>None</option>
                <option>1</option>
                <option>2</option>
                <option>3</option>
                <option>4 and above</option>
                </select></td>
</tr><tr>
        <th align=left>Number of bathroom(s):</th>
        <td><select size="1.5" name="bathroom">
                <option selected>None</option>
```

```
                <option>1</option>
                        <option>2</option>
                        <option>3 and above</option>
                        </select></td>
</tr><tr>
        <th align=left>Property built year:</th>
        <td><select size="1.5" name="builtyear">
                        <option selected>None</option>
                        <option>1995 - Current</option>
                        <option>1990 - 1994</option>
                        <option>1985 - 1989</option>
                        <option>1980 - 1984</option>
                        <option>1970 - 1979</option>
                        <option>Before 1970</option>
                        </select></td>
</tr><tr>
        <th align=left>Property price (US dollars):</th>
        <td><select size="1.5" name="price">
                        <option selected>None</option>
                        <option>Below $200,000</option>
                        <option>$200,000 - $299,999</option>
                        <option>$300,000 - $399,999</option>
                        <option>$400,000 - $499,999</option>
                        <option>$500,000 - $599,999</option>
                        <option>$600,000 - $699,999</option>
                        <option>$700,000 - $799,999</option>
                        <option>$800,000 and above</option>
                        </select></td>
</tr><tr>
        <th align=left>Property size (square feet):</th>
        <td><select size="1.5" name="size">
                        <option selected>None</option>
                        <option>Below 1,000</option>
                        <option>1,000 - 1,499</option>
                        <option>1,500 - 1,999</option>
                        <option>2,000 - 2,499</option>
                        <option>2,500 - 2,999</option>
                        <option>3,000 and above</option>
                        </select></td>
</tr>
</table>
<br>
<table>
<tr><td>
<input type="submit" value="Submit">
</td><td>
<input type="reset" value="Reset">
</td></tr>
</table>
</form>
</body>
</html>
```

```
<input type="submit" value="Submit">
</td><td></td></tr>
</table>
</form>
</body>
</html>
```

office.html

```
<html>
<head><title>Office Search</title></head>
<body>
<P><IMG SRC="images/office_searchB.gif" ALIGN=left><h2>Office Search</h2></P>
<table >
        <tr>
        <td>[ <a href="welcome.html">Home</a> ]</td>
        <td>[ <a href="mailto:webmaster@www.sweethome.com">Contact us</a> ]</td>
        </tr>
</table>
<hr size=5>

<form method="POST" action=branch.jsp>
 <p><input type="radio" value="B1" checked name="R1"> San Bernardino Office</p>
 <p><input type="radio" name="R1" value="B2"> Los Angeles Office</p>
 <p><input type="radio" name="R1" value="B3"> Hollywood Office</p>
 <p><input type="radio" name="R1" value="B4"> San Francisco Office</p>
 <p><input type="radio" name="R1" value="B5"> Ontario Office</p>
 <p><input type="radio" name="R1" value="B6"> Riverside Office</p>
 <p><input type="submit" value="Submit"> <input type="reset" value="Reset"></p>
</form>
</body>
</html>
```

app.html

```
<html>
<head><title>MAKE APPOINTMENT</title></head>

<body>
<table><tr><td><IMG SRC="images/handshake.jpg" width=75 height=100></td>
<td valign=bottom><H2><B>SET UP<P>APPOINTMENT</B></H2></td></tr></table><br>
<form method="POST" action="./servlet/makeAppServlet">

<table>
<tr><td width=100>Broker Id:</td><td width=150><input type="text" name="BID"
size="20"></td></tr>
<tr><td>Customer id:</td><td><input type="text" name="CID" size="20"></td></tr>
<tr><td>Date:</td><td>mm  <SELECT NAME="MONTH">
<OPTION VALUE="1">Jan</OPTION>
<OPTION VALUE="2">Feb</OPTION>
```

```
<OPTION VALUE="3">Mar</OPTION>
<OPTION VALUE="4">Apr</OPTION>
<OPTION VALUE="5">May</OPTION>
<OPTION VALUE="6">Jun</OPTION>
<OPTION VALUE="7">Jul</OPTION>
<OPTION VALUE="8">Aug</OPTION>
<OPTION VALUE="9">Sep</OPTION>
<OPTION VALUE="10">Oct</OPTION>
<OPTION VALUE="11">Nov</OPTION>
<OPTION VALUE="12">Dec</OPTION>
</SELECT>
dd  <SELECT NAME="DATE">
<OPTION VALUE="1">1</OPTION>
<OPTION VALUE="2">2</OPTION>
<OPTION VALUE="3">3</OPTION>
<OPTION VALUE="4">4</OPTION>
<OPTION VALUE="5">5</OPTION>
<OPTION VALUE="6">6</OPTION>
<OPTION VALUE="7">7</OPTION>
<OPTION VALUE="8">8</OPTION>
<OPTION VALUE="9">9</OPTION>
<OPTION VALUE="10">10</OPTION>
<OPTION VALUE="11">11</OPTION>
<OPTION VALUE="12">12</OPTION>
<OPTION VALUE="13">13</OPTION>
<OPTION VALUE="14">14</OPTION>
<OPTION VALUE="15">15</OPTION>
<OPTION VALUE="16">16</OPTION>
<OPTION VALUE="17">17</OPTION>
<OPTION VALUE="18">18</OPTION>
<OPTION VALUE="19">19</OPTION>
<OPTION VALUE="20">20</OPTION>
<OPTION VALUE="21">21</OPTION>
<OPTION VALUE="22">22</OPTION>
<OPTION VALUE="23">23</OPTION>
<OPTION VALUE="24">24</OPTION>
<OPTION VALUE="25">25</OPTION>
<OPTION VALUE="26">26</OPTION>
<OPTION VALUE="27">27</OPTION>
<OPTION VALUE="28">28</OPTION>
<OPTION VALUE="29">29</OPTION>
<OPTION VALUE="30">30</OPTION>
<OPTION VALUE="31">31</OPTION>
</SELECT>
yy <SELECT NAME="YEAR">
<OPTION VALUE="2002">2002</OPTION>
<OPTION VALUE="2003">2003</OPTION>
</SELECT>
</td></tr>
<tr><td valign=top>Time:</td>
<td><input type="radio" value="10:00" name="TIME" CHECKED>10:00 AM<BR>
   <input type="radio" value="14:00" name="TIME">02:00 PM</td></tr>
```

```
<tr><td valign=top>Purpose:</td>
<td><input type="radio" name="PURPOSE" value="BUY" CHECKED>Buy<BR>
<input type="radio" value="SELL" name="PURPOSE">Sell<BR>
<input type="radio" value="OTHER" name="PURPOSE">Other
</td></tr>
</table>
<p><input type="submit" value="Submit"> <input type="reset" value="Reset">
</form>
<p>[<a href="forgetid.html">Forget your ID?</a>]
[<a href="broker.html">View broker information</a>]
</body>
</html>
```

buy.html

```
<html>
<head><title>Property Buyer Guide</title></head>
<body>

<p><IMG SRC="images/handshake.jpg" ALIGN=left><H2>Property Buyer
Guide</H2></p><BR>
<p><BIG><I>Welcome! Choose one of the Following:</BIG></I></p><BR>
<BR>hello
<BR>
<p>
<p><A HREF="app.html"><IMG SRC="images/home_pageB.gif" ALIGN =left
BORDER=0></A><BR>
<A HREF="app.html">If you already know a broker and you
would like to make an appointment with him/her</A></p><BR>

<p><A HREF="ps.html"><IMG SRC="images/property_searchB.gif" ALIGN =left
BORDER=0></A><BR>
<A HREF="ps.html">Find a property matching you need</A></p><BR>

<p><A HREF="office.html"><IMG SRC="images/office_searchB.gif" ALIGN =left
BORDER=0></A><BR>
<A HREF="office.html">If you don't know a broker, and you
would like to search one by choosing a office near you</A></p><BR>

<p><A HREF="broker.html"><IMG SRC="images/broker_searchB.gif" ALIGN =left
BORDER=0></A><BR>
<A HREF="broker.html">If you don't know a broker, and you
would like to search one by broker's qualification</A></p><BR>
</body>
</html>
```

sell.html

```
<html>
<head><title>Property Owner Guide</title></head>
```

```
<body>
<p><IMG SRC="images/sold.gif" ALIGN=left><H2>Property Owner Guide</H2></p><BR>
<p><I><BOLD>Welcome! Choose one of the Following:</I></BOLD></p><BR>
<BR>
<p><A HREF="app.html"><IMG SRC="images/home_pageB.gif" ALIGN=left
BORDER=0></A><BR>
<A HREF="app.html">If you already know a broker and you
would like to make an appointment with him/her</A></p>
<BR>
<p><A HREF="office.html"><IMG SRC="images/office_searchB.gif" ALIGN=left
BORDER=0></A><BR>
<A HREF="office.html">If you don't know any broker, and you
would like to search one by choosing a office near you</A></p>
<BR>
<p><A HREF="broker.html"><IMG SRC="images/broker_searchB.gif" ALIGN=left
BORDER=0></A><BR>
<A HREF="broker.html">If you don't know any broker and you
would like to search one by broker's qualifications</A></p>
<BR>
</form>
</body>
</html>
```

forgetid.html

```
<html>
<head><title>Forget ID</title></head>
<body>

<br>
<form method="post" action="./servlet/forgetidServlet">
Answer the following questions to retrieve your ID:
<p>
<table>
<tr><td>SSN: </td><td><input type="text" name="SSN1" size="3"> -
                <input type="text" name="SSN2" size="2"> -
                <input type="text" name="SSN3" size="4"></td></tr>
<tr><td>Zip Code: </td><td><input type="text" name="ZIP" size="5"></td></tr>
</table>

<p><input type="submit" value="Submit"> <input type="reset" value="Reset"></p>
</form>
</body>
</html>
```

APPENDIX B

SOURCE CODE: JAVASERVER PAGE

PART

branch.jsp

```
<jsp:useBean id="officesearch" scope="session" class="sweethome.office"/>
<html>
<head><title>Office Search</title></head>

<%String branch = request.getParameter("R1");
officesearch.setbranchno(branch);%>
<body>

<h2>Office Search</h2>
<table >
        <tr>
        <td>[ <a href="welcome.html">Home</a> ]</td>
        <td>[ <a href="mailto:webmaster@www.sweethome.com">Contact us</a> ]</td>
        </tr>
</table>
<hr size=5>

<p><b><%=officesearch.information()[3]%> Office</b>

<table>
 <tr>
  <td><b>Address:</b>
  <td><%=officesearch.information()[2]%>

 <tr>
  <td><td>
   <%=officesearch.information()[3]%>,<%=officesearch.information()[4]%>
   <%=officesearch.information()[5]%>
 <tr>
  <td><b>Phone:</b>

<td>(<%=officesearch.information()[0].substring(0,3)%>)<%=officesearch.information()[0].subst
ring(3,10)%>
 <tr>
  <td><b>Fax:</b>

<td>(<%=officesearch.information()[1].substring(0,3)%>)<%=officesearch.information()[1].subst
ring(3,10)%>
 <tr>
  <td><b>Map:</b>
</table>
<img src="images/<%=branch%>.gif"><br>
<b>Manager:
<table>
 <%java.util.Vector managers = officesearch.manager();
                int i =0;
                                while(i<managers.size()){%>
 <tr>
  <td><%=managers.elementAt(i)%>,
   <%=managers.elementAt(++i)%>
```

```
   <td>   Phone
no:(<%=managers.elementAt(++i).toString().substring(0,3)%>)<%=managers.elementAt(i).toStri
ng().substring(3,10)%>
    <%i++;}%>
 </tr>
</table>
<a href="profile.jsp">Click here if
you would like to view sales profile of this office</a><br>
<a href="broker.jsp">Click here if
you would like to view brokers in this office</a></b>
</body>
</html>
```

Profile.jsp

```
<jsp:useBean id="officesearch" scope="session" class="sweethome.office"/>
<html>
<head><title>Office Profile</title></head>

<body>
<h2>Office Search</h2>
<table >
        <tr>
        <td>[ <a href="welcome.html">Home</a> ]</td>
        <td>[ <a href="mailto:webmaster@www.sweethome.com">Contact us</a> ]</td>
        </tr>
</table>
<hr size=5>
<p><%String branch=officesearch.getbranchno();%><h3><%=officesearch.information()[3]%>
Office</h3>
<p>Sales Profile:</p>
    #<%=officesearch.aggregate()[3]%>      properties sold this year.
<p>    Highest Property Sale Price:
$<%=officesearch.aggregate()[1]%>
</p>
<p>    Lowest Property Sale Price: $<%=officesearch.aggregate()[2]%>
<p>    Average Sale Price: $<%=officesearch.aggregate()[0]%>
<p><a href="office.html"><i>View
other office</i></a></p>
<p><i><a href="welcome.html">Go to
sweetHome</a></i></p>
</body>
</html>
```

broker.jsp

```
<jsp:useBean id="officesearch" scope="session" class="sweethome.office"/>
<html>
```

```
<head><title>Broker</title></head>

<body>
<h2>Office Search</h2>
<table >
        <tr>
        <td>[ <a href="welcome.html">Home</a> ]</td>
        <td>[ <a href="mailto:webmaster@www.sweethome.com">Contact us</a> ]</td>
        </tr>
</table>
<hr size=5>
<%String branch=officesearch.getbranchno();%><b><%=officesearch.information()[3]%>
Office</b>
<br>Broker list:</br>
<form method="POST" action="brokerprofile.jsp">
  <select size="10" name="BROKERID">
  <%java.util.Vector brokers = officesearch.brokers();%>
  <option value="<%=brokers.elementAt(0)%>" selected>
  <%=brokers.elementAt(1)%>,<%=brokers.elementAt(2)%></option>
  <%int i =3; while(i<brokers.size()){%>
  <option value="<%=brokers.elementAt(i)%>"><%=brokers.elementAt(++i)%>,
  <%=brokers.elementAt(++i)%></option><%i++;}%>
 </select>
 <p>To see the broker's profile, click- <input type="submit" value="See Profiles"></p>
 <p><br>
 </p>
</form>
</body>
</html>
```

brokerprofile.jsp

```
<jsp:useBean id="brokersearch" scope="session" class="sweethome.office"/>
<html>
<%String brokerid = request.getParameter("BROKERID");%>
<%brokersearch.setbrokerid(brokerid);%>
<head><title>Broker Profile</title></head>
<body>
<h2>Broker Profile</h2>
<table >
        <tr>
        <td>[ <a href="welcome.html">Home</a> ]</td>
        <td>[ <a href="mailto:webmaster@www.sweethome.com">Contact us</a> ]</td>
        </tr>
</table>
<hr size=5>

<form method="POST" action="./servlet/appServlet">
 <b>Broker ID: <%=brokerid%></b>        <input type="hidden" name="BROKERID"
value="<%=brokerid%>">
 <%if (brokersearch.sale()){%>
```

```
 <p>This broker sold <%=brokersearch.brokerprofile()[3]%>    properties this year
 <p>The average sale price is : $<%=brokersearch.brokerprofile()[0]%>
 <p>The highest sale price is : $<%=brokersearch.brokerprofile()[1]%>
 <p>The lowest sale price is : $<%=brokersearch.brokerprofile()[2]%>
 <%}else{%>
 <br>
 No sale record in year 2000.<%}%>
 <p>If you want to make an appointment with this broker, click- <input type="submit"
value="Make an appointment"><br>
 </p>
</form>
<p><a href="office.html"><b>View
other office</b></a></p>
<p><a href="welcome.html"><b>Go to
SweetHome</b></a></p>
</body>
</html>
```

APPENDIX C

SOURCE CODE: SERVLET AND

JAVABEAN PART

accessBean.java

```
import java.beans.*;
import java.sql.*;
import java.io.*;

public class accessBean
{  public accessBean(String db, String login, String pwd, String query)
   {  database = db;
      user = login;
      password = pwd;
      sql = query;
      resultArray = new String[getMaxResultArrayLen()];
      colArray = new String[getMaxColArrayLen()];
      numColumns = 0;
      fillStatus = false;
      changes = new PropertyChangeSupport(this);
   }

   public void execQuery()
   {  String url = "jdbc:mysql://localhost:3306/" + database;
      String driver = "org.gjt.mm.mysql.Driver";
      Connection con = null;
      java.sql.Statement stmt = null;
      ResultSet rs = null;
      ResultSetMetaData rsmd = null;
      String result;
      int index = 0;
      int numCols = 0;
      String colValue = null;
      try
      {  Class.forName(driver);
         con = DriverManager.getConnection(url, user, password);
         stmt = con.createStatement();
         rs = stmt.executeQuery(sql);
         rsmd = rs.getMetaData();
         numCols = rsmd.getColumnCount();
         setNumColumns(numCols);

         for(int i = 1; i <= numCols; i++)
         {  colValue = new String("");
            colValue = rsmd.getColumnLabel(i);
            setColArray(index, colValue);
            index++;
         }
         index = 0;
         while(rs.next())
         {  for(int i=1; i<=numCols;i++)
            {  result = rs.getString(i);
               setResultArray(index, result);
               index++;
            }
```

```
      }
      setFillStatus(true);
      rs.close();
      stmt.close();
      con.close();
    }
    catch(Exception e)
    { errMessage = e.getMessage();
      return;
    }
  }

  public void execUpdate()
  { String url = "jdbc:mysql://localhost:3306/" + database;
    String driver = "org.gjt.mm.mysql.Driver";
    Connection con = null;
    java.sql.Statement stmt = null;
    int index = 0;
    int numCols = 0;
    String colValue = null;
    try
    { Class.forName(driver);
      con = DriverManager.getConnection(url, user, password);
      stmt = con.createStatement();
      stmt.executeUpdate(sql);
      setFillStatus(true);
      stmt.close();
      con.close();
    }
    catch(Exception e)
    { errMessage = e.getMessage();
      return;
    }
  }

  public void addPropertyChangeListener(PropertyChangeListener l)
  { changes.addPropertyChangeListener(l);
  }

  public void removePropertyChangeListener(PropertyChangeListener l)
  { changes.removePropertyChangeListener(l);
  }

  public synchronized boolean getFillStatus()
  { return fillStatus;
  }

  public void setFillStatus(boolean newValue)
  { boolean oldValue = fillStatus;
    fillStatus = newValue;
    Boolean oldObj = new Boolean(oldValue);
    Boolean newObj = new Boolean(newValue);
```

```
  changes.firePropertyChange("fillStatus", oldObj, newObj);
}

public String getColArray(int index)
{ return this.colArray[index];
}

public void setColArray(int index, String value)
{ this.colArray[index] = value;
}

public String getResultArray(int index)
{ return this.resultArray[index];
}

public void setResultArray(int index, String value)
{ this.resultArray[index] = value;
}

public void setNumColumns(int n)
{ numColumns = n;
}

public int getNumColumns()
{ return numColumns;
}

public int getMaxResultArrayLen()
{ return maxResultArrayLen;
}

public void setMaxResultArrayLen(int i)
{ maxResultArrayLen = i;
}

public int getMaxColArrayLen()
{ return maxColArrayLen;
}

public void setMaxColArrayLen(int i)
{ maxColArrayLen = i;
}

public String getErrorMsg()
{ return errMessage;
}

// bean properties
private String database;
private String user;
private String password;
private String sql;
```

```
  // bean properties associated with query results.
  private String[] resultArray = null;
  private String[] colArray = null;
  private int maxResultArrayLen = 1000;
  private int maxColArrayLen = 30;
  private boolean fillStatus = false;
  private int numColumns;
  private String errMessage = null;

  // used for firing events
  private PropertyChangeSupport changes = null;
}
```

PropertyInfo.java

```
/* This class is used to stored information entered by users. Its life cycle is
  * transient as long as the session for that user exists.*/

public class PropertyInfo
{       private String city = "";
        private String zip = "";
        private String propertyId = "";
        private String type = "";
        private   String bedroom = "";
        private   String bathroom = "";
        private String builtyear = "";
        private   String price = "";
        private   String size = "";
        private String errMsg = "";

        public void setCity(String city)
        {       this.city = city;
        }

        public String getCity()
        {       return city;
        }

        public void setZip(String zip)
        {       this.zip = zip;
        }

        public String getZip()
        {       return zip;
        }

        public void setPropertyId(String id)
        {       propertyId = id;
        }

        public String getPropertyId()
```

```
{       return propertyId;
}

public void setType(String type)
{       this.type = type;
}

public String getType()
{       return type;
}

public void setBedroom(String bedroom)
{       this.bedroom = bedroom;
}

public String getBedroom()
{       return bedroom;
}

public void setBathroom(String bathroom)
{       this.bathroom = bathroom;
}

public String getBathroom()
{       return bathroom;
}

public void setBuiltyear(String builtyear)
{       this.builtyear = builtyear;
}

public String getBuiltyear()
{       return builtyear;
}

public void setPrice(String price)
{       this.price = price;
}

public String getPrice()
{       return price;
}

public void setSize(String size)
{       this.size = size;
}

public String getSize()
{       return size;
}

public void setErrMsg(String msg)
```

```
        {       errMsg += msg + '\n';
        }

        public String getErrMsg()
        {       return errMsg;
        }

        public void reset()
        {       city = "";
                zip = "";
                propertyId = "";
                type = "";
                bedroom = "";
                bathroom = "";
                builtyear = "";
                price = "";
                size = "";
        }
}
```

BrokerInfo.java

```
public class BrokerInfo
{private String language = "";
 private String year = "";
 private String city = "";
 private String numofsales = "";

 public void setLanguage(String language)
 {  this.language = language;
 }

 public String getLanguage()
 {  return language;
 }

 public void setYear(String year)
 {  this.year = year;
 }

 public String getYear()
 {  return year;
 }

 public void setCity(String city)
 {  this.city = city;
 }

 public String getCity()
 {  return city;
 }
```

```
 public void setNumofSales(String numofsales)
 {   this.numofsales = numofsales;
 }

 public String getNumofSales()
 {   return numofsales;
 }

 public void reset()
 {language = "";
  year = "";
  city = "";
  numofsales = "";
 }
}
```

office.java

```
package sweethome;
import java.io.*;
import java.lang.*;
import java.util.*;

public class office{
   private String branchno;
   private String brokerid;
   private String userid = "root";
   private String passwd = "123456";
   public accessBean aB = null;
   public office()
   { }

   public String[] information(){
    String sql = "SELECT TELNO,FAXNO,STREET,CITY,STATE,CODE FROM BRANCH
                  WHERE BRANCHNO ='"+branchno+"'";
     aB = new accessBean("sweethome", userid, passwd, sql);
     aB.execQuery();
     int numElements = aB.getMaxResultArrayLen();
     int currentElement = 0;
     String[] rs = new String[numElements];

    while(currentElement < numElements &&
        aB.getResultArray(currentElement) != null)
    {    rs[currentElement] = aB.getResultArray(currentElement).trim();
         currentElement++;
     }
     return rs;
   }

   public Vector manager(){
    String sql = "select lname,fname,workno from branch,duration,employee where
```

```
                employee.branchno = branch.branchno and branch.branchno ='"+branchno+"'";
    aB = new accessBean("sweethome", userid, passwd, sql);
     aB.execQuery();
     int numElements = aB.getMaxResultArrayLen();
     int currentElement = 0;
     Vector rs = new Vector();
     while(currentElement < numElements &&
         aB.getResultArray(currentElement) != null)
         {rs.addElement(aB.getResultArray(currentElement).trim());
          currentElement++;
         }
         return rs;
   }

   public int[] aggregate(){
     String sql="SELECT AVG(TRANSAMOUNT), MAX(TRANSAMOUNT),
                 MIN(TRANSAMOUNT), COUNT(TRANSNO) FROM SALERECORD
                 WHERE BRANCHNO ='"+branchno+"' and year(transdate)=2002 GROUP BY
                 BRANCHNO";
     aB = new accessBean("sweethome", userid, passwd, sql);
     aB.execQuery();
     int[] rs=new int[4];
     for(int i =0;i<4;i++){
     rs[i] = Float.valueOf(aB.getResultArray(i).trim()).intValue();
     }
     return rs;
   }
    public Vector brokers(){
    String sql = "select brokerid,lname,fname from employee,broker where
broker.ssn=employee.ssn and branchno='"+branchno+"'";
    aB = new accessBean("sweethome", userid, passwd, sql);
    aB.execQuery();
     int numElements = aB.getMaxResultArrayLen();
     int currentElement = 0;
     Vector rs = new Vector();
     while(currentElement < numElements &&
         aB.getResultArray(currentElement) != null)
         {rs.addElement(aB.getResultArray(currentElement).trim());
          currentElement++;
         }
         return rs;
   }

   public boolean sale(){
     String sql="SELECT BROKERID FROM SALERECORD WHERE BROKERID
                 ='"+brokerid+"' AND YEAR(TRANSDATE)=2002";
     aB = new accessBean("sweethome", userid, passwd, sql);
     aB.execQuery();
     if(aB.getResultArray(0)!=null)
     return true;
     else
     return false;
```

```
    }
    public int[] brokerprofile(){
      String sql="SELECT AVG(TRANSAMOUNT), MAX(TRANSAMOUNT),
                  MIN(TRANSAMOUNT), COUNT(TRANSNO) FROM SALERECORD
                  WHERE BROKERID ='"+brokerid+"' and year(transdate)=2002 GROUP BY
                  BROKERID";
      aB = new accessBean("sweethome", userid, passwd, sql);
      aB.execQuery();
      int[] rs=new int[4];
      for(int i =0;i<4;i++){
      rs[i] = Float.valueOf(aB.getResultArray(i).trim()).intValue();
      }
      return rs;
    }

    public int brokerexp(){
      String sql="SELECT year(DATETIMEEXPSTART) FROM BROKER WHERE
                  BROKERID ='"+brokerid+"'";
      aB = new accessBean("sweethome", userid, passwd, sql);
      aB.execQuery();
      int rs = 2002- Float.valueOf(aB.getResultArray(0).trim()).intValue();
      return rs;
    }

    public void setbranchno(String bno) {
      branchno = bno;
    }

    public String getbranchno() {
      return branchno;
    }

    public void setbrokerid(String bid) {
      brokerid = bid;
    }

    public String getbrokerid() {
      return brokerid;
    }
}
```

psCheck.java

```
/* This servlet checks user's submit information from ps.html. If information is valid,
  * the request will be forwarded to psQuery servlet, if information is not valid,
  * the request will be fowarded to psAgain servlet.  */

import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
```

```
public class psCheck extends HttpServlet
{
  public void doPost(HttpServletRequest req, HttpServletResponse res)
                                        throws IOException, ServletException
  {         // Get user's session and property info
            HttpSession session = req.getSession(true);
            PropertyInfo pInfo = (PropertyInfo)session.getValue(session.getId());
            // If the user has no proeprty info object, create a new one
            if(pInfo == null)
            {       pInfo = new PropertyInfo();
                    session.putValue(session.getId(), pInfo);
            }
            // Gather property serach information
            String city = req.getParameter("city");
            String zip = req.getParameter("zip");
            String id = req.getParameter("id");
            String house = req.getParameter("house");
            String bedroom = req.getParameter("bedroom");
            String bathroom = req.getParameter("bathroom");
            String builtyear = req.getParameter("builtyear");
            String price = req.getParameter("price");
            String size = req.getParameter("size");
            city = city.trim();
            zip = zip.trim();
            id = id.trim();
            // Error checking
            if(city.equals("") && zip.equals("") && id.equals(""))
            {       pInfo.setErrMsg("Error: You have to specify a location for search or"
                                    + " a valid property id.");
                    RequestDispatcher dispatcher =
                            getServletContext().getRequestDispatcher("/servlet/psAgain");

                    if (dispatcher == null)
                    {       // No dispatcher means the html file can not be delivered
                            res.sendError(res.SC_NO_CONTENT);
                    }
                    dispatcher.forward(req, res);
            }
            else
            {       // Information OK. Set information in  and send to psOutput servlet
                    pInfo.setCity(city);
                    pInfo.setZip(zip);
                    pInfo.setPropertyId(id);
                    pInfo.setType(house);
                    pInfo.setBedroom(bedroom);
                    pInfo.setBathroom(bathroom);
                    pInfo.setBuiltyear(builtyear);
                    pInfo.setPrice(price);
                    pInfo.setSize(size);
                    RequestDispatcher dispatcher =
                            getServletContext().getRequestDispatcher("/servlet/psQuery");
```

```
                    if (dispatcher == null)
                    {       // No dispatcher means the html file can not be delivered
                            res.sendError(res.SC_NO_CONTENT);
                    }
                    dispatcher.forward(req, res);
                }
        }

        public String getServletInfo()
        {       return "A servlet that checks user's input information and forwards request"
                        + "to appropriate servlets";
        }
}
```

psQuery.java

```
/* This servlet displays property information based on property information user entered
  * in ps.html.  */

import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class psQuery extends HttpServlet
{       private String url;
        private String driver;
        private accessBean aBean;
        private String database;
        private String login;
        private String password;

        public void init()
        { database = "sweethome";
          login = "root";
          password = "123456";
        }

   public void doPost(HttpServletRequest req, HttpServletResponse res)
                                                throws IOException, ServletException
   {            String type = "";
                String city = "";
                String zip = "";
                String id = "";
                String bedroom = "";
                String bathroom  = "";
                String builtyear = "";
                String price = "";
                String size = "";
                // Get user's session and DB info
                HttpSession session = req.getSession(true);
                PropertyInfo pInfo = (PropertyInfo)session.getValue(session.getId());
```

```
if(pInfo == null)
{  // If the user has no property information object,
   //foward to an error pbuiltyear
 RequestDispatcher dispatcher =
  getServletContext().getRequestDispatcher("/servlet/psErrSessionPage");
        if (dispatcher == null)
        {  // No dispatcher means the html file can not be delivered
              res.sendError(res.SC_NO_CONTENT);
        }
        dispatcher.forward(req, res);
}
else
{       // get user input property information
        type = pInfo.getType();
        city = pInfo.getCity();
        zip = pInfo.getZip();
        id = pInfo.getPropertyId();
        bedroom = pInfo.getBedroom();
        bathroom = pInfo.getBathroom();
        builtyear = pInfo.getBuiltyear();
        price = pInfo.getPrice();
        size = pInfo.getSize();
}

PrintWriter out = res.getWriter();
res.setContentType("text/html");
out.println("<html>");
out.println("<head><title>Property Search Result</title></head>");
out.println("<body>");
out.println("<form method=\"post\""          + "action=\"./appServlet\">");
out.println("<h2>Property Search Results</h2>");
out.println("<table>");
out.println("<tr>");
out.println("<td>[ <a href=\"/kevin-p/welcome.html\">Home</a> ]</td>");
out.println("<td>[ <a href=\"mailto:webmaster@sweethome.com\">Contact
                                                    us</a>"  + "]</td>");
out.println("</tr></table>");
out.println("<hr size=5>");
out.println("<br>");
out.println("<table width=80%>");
String where = "where ";

if(!city.equals(""))
        where += "lower(p.city) = lower('" + city + "') AND ";

if(!zip.equals(""))
        where += "p.code = '" + zip + "' AND ";

if(!id.equals(""))
        where += "lower(p.propertyid) = lower('" + id + "') AND ";

if(type.equalsIgnoreCase("Single family house"))
```

```
        where += "p.type = 'Single family house' AND ";
else if(type.equals("Town house"))
        where += "p.type = 'Town house' AND ";
else if(type.equals("Condo"))
        where += "p.type = 'Condo' AND ";

if(bedroom.equals("1"))
        where += "p.noroom = 1 AND ";
else if(bedroom.equals("2"))
        where += "p.noroom = 2 AND ";
else if(bedroom.equals("3"))
        where += "p.noroom = 3 AND ";
else if(bedroom.equalsIgnoreCase("4 and above"))
        where += "p.noroom >= 4 AND ";

if(bathroom.equals("1"))
        where += "p.nobath = 1 AND ";
else if(bathroom.equals("2"))
        where += "p.nobath = 2 AND ";
else if(bathroom.equalsIgnoreCase("3 and above"))
        where += "p.nobath >= 3 AND ";

if(builtyear.equalsIgnoreCase("1995 - Current"))
        where += "p.builtyear >= 1995 AND ";
else if(builtyear.equalsIgnoreCase("1990 - 1994"))
        where += "p.builtyear >= 1990 AND p.builtyear <= 1994 AND ";
else if(builtyear.equalsIgnoreCase("1985 - 1989"))
        where += "p.builtyear >= 1985 AND p.builtyear <= 1989 AND ";
else if(builtyear.equalsIgnoreCase("1980 - 1984"))
        where += "p.builtyear >= 1980 AND p.builtyear <= 1984 AND ";
else if(builtyear.equalsIgnoreCase("1970 - 1979"))
        where += "p.builtyear >= 1970 AND p.builtyear <= 1979 AND ";
else if(builtyear.equalsIgnoreCase("Before 1970"))
        where += "p.builtyear <= 1970 AND ";

if(price.equalsIgnoreCase("Below $200,000"))
        where += "p.price <= 200000 AND ";
else if(price.equalsIgnoreCase("$200,000 - $299,999"))
        where += "p.price >= 200000 AND p.price <= 299999 AND ";
else if(price.equalsIgnoreCase("$300,000 - $399,999"))
        where += "p.price >= 300000 AND p.price <= 399999 AND ";
else if(price.equalsIgnoreCase("$400,000 - $499,999"))
        where += "p.price >= 400000 AND p.price <= 499999 AND ";
else if(price.equalsIgnoreCase("$500,000 - $599,999"))
        where += "p.price >= 500000 AND p.price <= 599999 AND ";
else if(price.equalsIgnoreCase("$600,000 - $699,999"))
        where += "p.price <= 600000 AND p.price >= 699999 AND ";
else if(price.equalsIgnoreCase("$700,000 - $799,999"))
        where += "p.price <= 700000 AND p.price >= 799999 AND ";
else if(price.equalsIgnoreCase("$800,000 and above"))
        where += "p.price >= 800000 AND ";
```

```
if(size.equalsIgnoreCase("Below 1,000"))
        where += "p.squareft <= 1000 AND ";
else if(size.equalsIgnoreCase("1,000 - 1,499"))
        where += "p.squareft >= 1000 AND p.squareft <= 1499 AND ";
else if(size.equalsIgnoreCase("1,500 - 1,999"))
        where += "p.squareft >= 1500 AND p.squareft <= 1999 AND ";
else if(size.equalsIgnoreCase("2,000 - 2,499"))
        where += "p.squareft >= 2000 AND p.squareft <= 2499 AND ";
else if(size.equalsIgnoreCase("2,500 - 2,999"))
        where += "p.squareft >= 2500 AND p.squareft<= 2999 AND ";
else if(size.equalsIgnoreCase("3,000 and above"))
        where += "p.squareft >= 3000 AND ";

String sql = "SELECT p.propertyid, p.type, p.street, p.city, p.state, p.code, "
             + " p.noroom, p.nobath, p.squareft, p.builtyear, p.price, o.brokerid "
             + " FROM property p, propertyowned po, owner o " + where
             + " p.propertyid=po.propertyid AND po.currentowned='Y' "
             + " AND po.ssn=o.ssn ";
aBean = new accessBean(database, login, password, sql);
aBean.execQuery();

if(!aBean.getFillStatus())
{       out.println("<p>No matched properties have been
                                                found</p><br> ");
        out.println("<a href=\"/kevin-p/ps.html\">Go to Property
                    Search</a><br>");
        out.println("<a href=\"/kevin-p/welcome.html\">Go to
                    Sweethome</a>");
}
else
{       int numCol = aBean.getNumColumns();
        String propertyId;
        String street;
        String state;
        String code;
        String noRoom;
        String noBath;
        String squareFt;
        String builtYear;
        String brokerId;
        String result;
        int index=0;

        if(aBean.getResultArray(index)!=null)
        {       while((propertyId=aBean.getResultArray(index++))!=null)
                {       type = aBean.getResultArray(index++);
                        street = aBean.getResultArray(index++);
                        city = aBean.getResultArray(index++);
                        state = aBean.getResultArray(index++);
                        code = aBean.getResultArray(index++);
                        noRoom = aBean.getResultArray(index++);
```

```
                noBath = aBean.getResultArray(index++);
                squareFt = aBean.getResultArray(index++);
                builtYear = aBean.getResultArray(index++);
                price = aBean.getResultArray(index++);
                brokerId = aBean.getResultArray(index++);
                out.println("<tr><td><input type=radio
                        name=\"BROKERID\" value=\""
                                + brokerId + "\"> ");
                out.println("</td><th align=left>");
                out.println("Property ID:");
                out.println("</th><td align=left>");
                out.println(propertyId);
                out.println("</td><tr><td></td><th align=left>");
                out.println("Address:");
                out.println("</th><td align=left>");
                out.println(street + ", " + city + ", " + state + " " +
                        code);
                out.println("</td><tr><td></td><th align=left>");
                out.println("Type:");
                out.println("</th><td align=left>");
                out.println(type);
                out.println("<br></td><tr><td></td><th
                        align=left>");
                out.println("No. of bedroom(s):");
                out.println("</th><td align=left>");
                out.println(noRoom);
                out.println("</td><tr><td></td><th align=left>");
                out.println("No. of bathroom(s):");
                out.println("</th><td align=left>");
                out.println(noBath);
                out.println("</td><tr><td></td><th align=left>");
                out.println("Size:");
                out.println("</th><td align=left>");
                out.println(squareFt + " square feet");
                out.println("</td><tr><td></td><th align=left>");
                out.println("Built year:");
                out.println("</th><td align=left>");
                out.println(builtYear);
                out.println("</td><tr><td></td><th align=left>");
                out.println("Price:");
                out.println("</th><td align=left>");
                out.println("$" + price + " USD");
                out.println("</td><tr><td></td><th align=left>");
                out.println("Broker ID:");
                out.println("</th><td align=left>");
                out.println(brokerId);
                out.println("</td></tr><tr></tr><tr></tr>");
        }
        out.println("</table>");
}
else
{       out.println("<p>No matched records found</p>");
```

```
                    }
                }

                out.println("<br> ");
                out.println("<p>If you are not registered with us previously, " + "please go to
                        <a href=\"/kevin-p/cr.html\">Customer Registration"+ "</a>
                        before you make an appointment with our broker(s).");
                out.println("<p>If you are interested in one of the listed properties, "
                        + "select the property and click on \"Make an appointment\" "
                        + "button to make an appointment with our broker.</p>");
                out.println("<input type=\"submit\" value=\"Make an appointment\">");
                out.println("</form>");
                out.println("</body></html>");
                out.close();
                pInfo.reset();
                session.invalidate();
    }

    public String getServletInfo()
    {           return "A servlet that perform property search on the sweethome database";
        }
}
```

psAgain.java

```
/* This servlet displays error messbuiltyears and re-prompt users for property information.  */

import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class psAgain extends HttpServlet
{   public void doPost(HttpServletRequest req, HttpServletResponse res)
                                                throws IOException, ServletException
    {       res.setContentType("text/html");
            PrintWriter out = res.getWriter();
            out.println("<html>");
            out.println("<head><title>Property Search</title></head>");
            out.println("<body>");
                    // Get user's session and property info
                    HttpSession session = req.getSession(true);
                    PropertyInfo pInfo =  (PropertyInfo)session.getValue(session.getId());
                    out.println("<h2>Re-Enter Property Infomration</h2>");
                    out.println("<table>");
                    out.println("<tr>");
                    out.println("<td>[ <a href=\"welcome.html\">Home</a> ]</td>");
                    out.println("<td>[ <a href=\"mailto:webmaster@sweethome.com\">Contact
                                us</a> ]</td>");
                    out.println("</tr></table>");
                    out.println("<hr size=5>");
                    out.println("<br>");
```

```
// If the user has no proeprty info object, displays error messbuiltyear
if(pInfo == null)
{                   out.println("<p>Error on session. Please re-enter property
                        information</p>");
}
else
{                   out.println("<p>" + pInfo.getErrMsg() + "</p><br>");
}
out.println("<form method=\"post\" action=\"./psCheck\">");
out.println("<h2>Location for Search</h2>");
out.println("<table>");
out.println("<tr>");
out.println("<th align=left>(Option 1) </th>");
out.println("<th></th>");
out.println("<th align=left>Enter a city:</th>");
out.println("<td><input type=\"text\" name=\"city\" size=\"20\"></td>");
out.println("</tr><tr>");
out.println("<th align=left>(Option 2) </th>");
out.println("<th></th>");
out.println("<th align=left>Enter a zip code:</th>");
out.println("<td><input type=\"text\" name=\"zip\" size=\"20\"></td>");
out.println("</tr><tr>");
out.println("<th align=left>(Option 3) </th>");
out.println("<th></th>");
out.println("<th align=left>Property id (if known):</th>");
out.println("<td><input type=\"text\" name=\"id\" size=\"20\"></td>");
out.println("</tr>");
out.println("</table>");
out.println("<br>");
out.println("<h2>Additional Information</h2>");
out.println("<table border=1>");
out.println("<tr><th align=left>Property type:</th>");
out.println("<td><select size=\"1.5\" name=\"house\">");

String type = pInfo.getType();

if(type.equals("None"))
        out.println("<option selected>None</option>");
else
        out.println("<option>None</option>");

if(type.equals("Single family house"))
        out.println("<option selected>Single family house</option>");
else
        out.println("<option>Single family house</option>");

if(type.equals("Town house"))
        out.println("<option selected>Town house</option>");
else
        out.println("<option>Town house</option>");

if(type.equals("Condo"))
```

```
            out.println("<option selected>Condo</option>");
else
            out.println("<option>Condo</option>");

out.println("</select></td>");
out.println("</tr><tr>");
out.println("<th align=left>Number of bedroom(s):</th>");
out.println("<td><select size=\"1.5\" name=\"bedroom\">");

String bedroom = pInfo.getBedroom();

if(bedroom.equals("None"))
            out.println("<option selected>None</option>");
else
            out.println("<option>None</option>");

if(bedroom.equals("1"))
            out.println("<option selected>1</option>");
else
            out.println("<option>1</option>");

if(bedroom.equals("2"))
            out.println("<option selected>2</option>");
else
            out.println("<option>2</option>");

if(bedroom.equals("3"))
            out.println("<option selected>1</option>");
else
            out.println("<option>3</option>");

if(bedroom.equals("4 and above"))
            out.println("<option selected>4 and above</option>");
else
            out.println("<option>4 and above</option>");

out.println("</select></td>");
out.println("</tr><tr>");
out.println("<th align=left>Number of bathroom(s):</th>");
out.println("<td><select size=\"1.5\" name=\"bathroom\">");

String bathroom = pInfo.getBathroom();

if(bathroom.equals("None"))
            out.println("<option selected>None</option>");
else
            out.println("<option>None</option>");

if(bathroom.equals("1"))
            out.println("<option selected>1</option>");
else
            out.println("<option>1</option>");
```

```
if(bathroom.equals("2"))
        out.println("<option selected>2</option>");
else
        out.println("<option>2</option>");

if(bathroom.equals("3 and above"))
        out.println("<option selected>3 and above</option>");
else
        out.println("<option>3 and above</option>");

out.println("</select></td>");
out.println("</tr><tr>");
out.println("<th align=left>builtyear of the property (years):</th>");
out.println("<td><select size=\"1.5\" name=\"builtyear\">");
String builtyear = pInfo.getBuiltyear();

if(builtyear.equals("None"))
                out.println("<option selected>None</option>");
else
        out.println("<option>None</option>");

if(builtyear.equals("1995 - Current"))
        out.println("<option selected>1995 - Current</option>");
else
        out.println("<option>1995 - Current</option>");

if(builtyear.equals("1990 - 1994"))
        out.println("<option selected>1990 - 1994</option>");
else
        out.println("<option>1990 - 1994</option>");

if(builtyear.equals("1985 - 1989"))
        out.println("<option selected>1985 - 1989</option>");
else
        out.println("<option>1985 - 1989</option>");

if(builtyear.equals("1980 - 1984"))
        out.println("<option selected>1980 - 1984</option>");
else
        out.println("<option>1980 - 1984</option>");

if(builtyear.equals("1970 - 1979"))
        out.println("<option selected>1970 - 1979</option>");
else
        out.println("<option>1970 - 1979</option>");

if(builtyear.equals("Before 1970"))
        out.println("<option selected>Before 1970</option>");
else
        out.println("<option>Before 1970</option>");

out.println("</select></td>");
```

```
out.println("</tr><tr>");
out.println("<th align=left>Property price (US dollars):</th>");
out.println("<td><select size=\"1.5\" name=\"price\">");
String price = pInfo.getPrice();

if(price.equals("None"))
        out.println("<option selected>None</option>");
else
        out.println("<option>None</option>");

if(price.equals("Below $200,000"))
        out.println("<option selected>Below $200,000</option>");
else
        out.println("<option>Below $200,000</option>");

if(price.equals("$200,000 - $299,999"))
        out.println("<option selected>$200,000 - $299,999</option>");
else
        out.println("<option>$200,000 - $299,999</option>");

if(price.equals("$200,000 - $299,999"))
        out.println("<option selected>$200,000 - $299,999</option>");
else
        out.println("<option>$200,000 - $299,999</option>");

if(price.equals("$300,000 - $399,999"))
        out.println("<option selected>$300,000 - $399,999</option>");
else
        out.println("<option>$300,000 - $399,999</option>");

if(price.equals("$400,000 - $499,999"))
        out.println("<option selected>$400,000 - $499,999</option>");
else
        out.println("<option>$400,000 - $499,999</option>");

if(price.equals("$500,000 - $599,999"))
        out.println("<option selected>$500,000 - $599,999</option>");
else
        out.println("<option>$500,000 - $599,999</option>");

if(price.equals("$600,000 - $699,999"))
        out.println("<option selected>$600,000 - $699,999</option>");
else
        out.println("<option>$600,000 - $699,999</option>");

if(price.equals("$700,000 - $799,999"))
        out.println("<option selected>$700,000 - $799,999</option>");
else
        out.println("<option>$700,000 - $799,999</option>");

if(price.equals("$800,000 and above"))
        out.println("<option selected>$800,000 and above</option>");
```

```
else
        out.println("<option>$800,000 and above</option>");

out.println("</select></td>");
out.println("</tr><tr>");
out.println("<th align=left>Property size (square feet):</th>");
out.println("<td><select size=\"1.5\" name=\"size\">");
String size = pInfo.getSize();

if(size.equals("None"))
        out.println("<option selected>None</option>");
else
        out.println("<option>None</option>");

if(size.equals("Below 1,000"))
        out.println("<option selected>Below 1,000</option>");
else
        out.println("<option>Below 1,000</option>");

if(size.equals("1,000 - 1,499"))
        out.println("<option selected>1,000 - 1,499</option>");
else
        out.println("<option>1,000 - 1,499</option>");

if(size.equals("1,500 - 1,999"))
        out.println("<option selected>1,500 - 1,999</option>");
else
        out.println("<option>1,500 - 1,999</option>");

if(size.equals("2,000 - 2,499"))
        out.println("<option selected>2,000 - 2,499</option>");
else
        out.println("<option>2,000 - 2,499</option>");

if(size.equals("2,500 - 2,999"))
        out.println("<option selected>2,500 - 2,999</option>");
else
        out.println("<option>2,500 - 2,999</option>");

if(size.equals("3,000 and above"))
        out.println("<option selected>3,000 and above</option>");
else
        out.println("<option>3,000 and above</option>");

out.println("</select></td>");
out.println("<<tr>");
out.println("</table>");
out.println("<br> ");
out.println("<table ><tr><td>");
out.println("<input type=\"submit\" value=\"Submit\">");
out.println("</td><td>");
out.println("<input type=\"reset\" value=\"Reset\">");
```

```
                out.println("</td></tr></table>");
                out.println("</form>");
                out.println("</body>");
                out.println("</html>");
        }

        public String getServletInfo()
        {       return "A servlet that re-prompt user for property inforamtion";
        }
}
```

bsCheck.java

```
/* This servlet checks user's submit information from broker.html. If information is valid,
 * the request will be forwarded to bsQuery servlet. */

import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class bsCheck extends HttpServlet
{   public void doPost(HttpServletRequest req, HttpServletResponse res)
                                                throws IOException, ServletException
  {       // Get user's session and property info
                HttpSession session = req.getSession(true);
                BrokerInfo bInfo =  (BrokerInfo)session.getValue(session.getId());
    // If the user has no broker info object, create a new one
                if(bInfo == null)
                {       bInfo = new BrokerInfo();
                        session.putValue(session.getId(), bInfo);
                }
                // Gather property serach information
                String language = req.getParameter("language");
                String year = req.getParameter("year");
                String city = req.getParameter("city");
                String numofsales = req.getParameter("numofsales");
                // Error checking
          //Information OK. Set information in and send to psOutput servlet
  bInfo.setLanguage(language);
  bInfo.setYear(year);
  bInfo.setCity(city);
  bInfo.setNumofSales(numofsales);
  RequestDispatcher dispatcher =  getServletContext().getRequestDispatcher("/servlet/bsQuery");

    if (dispatcher == null)
    { // No dispatcher means the html file can not be delivered
      res.sendError(res.SC_NO_CONTENT);
    }
  dispatcher.forward(req, res);
 }
        public String getServletInfo()
```

```
        {       return "A servlet that checks user's input information and forwards request"
                        + "to appropriate servlets";
        }
}
```

bsQuery.java

```
/* The list of the broker search result.  */

import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class bsQuery extends HttpServlet
{       private String url;
        private String driver;
        private accessBean aBean;
        private String database;
        private String login;
        private String password;

        public void init()
        { database = "sweethome";
          login = "root";
          password = "123456";
        }

 public void doPost(HttpServletRequest req, HttpServletResponse res)
                                                throws IOException, ServletException
 {              String language1 = "";
                String year = "";
                String city = "";
                String numofsales = "";
  // Get user's session and DB info
  HttpSession session = req.getSession(true);
  BrokerInfo bInfo = (BrokerInfo)session.getValue(session.getId());
        PrintWriter out = res.getWriter();
        res.setContentType("text/html");
        out.println("<html>");
        out.println("<head><title>Broker Search Result</title></head>");
        out.println("<body>");
        out.println("<form method=\"post\""          + "action=\"/servlet/appServlet\">");
        out.println("<h2>Broker Search Results</h2>");
        out.println("<table>");
        out.println("<tr>");
        out.println("<td>[ <a href='../welcome.html'>Home</a> ]</td>");
        out.println("<td>[ <a href=\"mailto:webmaster@sweethome.com\">Contact us</a>"
                                + "]</td>");
        out.println("</tr></table>");
        out.println("<hr size=5>");
        out.println("<br> ");
```

```
        out.println("<table width=60%>");
        language1 = bInfo.getLanguage();
          year = bInfo.getYear();
          city = bInfo.getCity();
          numofsales = bInfo.getNumofSales();
        String where = "where";

//get broker language
where += " b.brokerid in (select brokerid from brokerlanguage where language= '"
                + language1 + "')";

//get year of experience
                if(year.equals("0-10"))
                where += " and b.brokerid in (select brokerid from broker b, employee e"
                + " where b.ssn=e.ssn AND ((to_days(now())-to_days(yearexpstart))/365)<11"
                + " AND  ((to_days(now())-to_days(yearexpstart))/365)>=0)";

                else if(year.equals("11-20"))
                where += " and b.brokerid in (select brokerid from broker b, employee e"
                + " where b.ssn=e.ssn AND ((to_days(now())-to_days(yearexpstart))/365)<21"
                + " AND  ((to_days(now())-to_days(yearexpstart))/365)>=11)";

                 else if(year.equals("21 and above"))
                where += " and b.brokerid in (select brokerid from broker b, employee e"
                + " where b.ssn=e.ssn AND ((to_days(now())-to_days(yearexpstart))/365)>21)";

//get location of office
where += " and b.brokerid in (select brokerid from broker b, employee e, branch r"
          + " where b.ssn=e.ssn and e.branchno=r.branchno and r.city= '" + city + "') ";

//get # of sales
if(numofsales.equals("0-10"))
where += " and b.brokerid in (select brokerid from salerecord group by brokerid"
          + " having count(*)>=0 AND count(*)<=10)";

else if(numofsales.equals("11 and above"))
       where += " and b.brokerid in (select brokerid from salerecord group by brokerid"
                 + " having count(*)>=11)";

//Query for broker information
String sql = "SELECT b.brokerid, e.lname, e.fname  "+ " FROM broker b, employee e " +
                Where + " AND b.ssn = e.ssn";
aBean = new accessBean(database, login, password, sql);
aBean.execQuery();
String brokerId;
String lname;
String fname;
int index=0;
while((brokerId=aBean.getResultArray(index++))!=null)
{lname = aBean.getResultArray(index++);
 fname = aBean.getResultArray(index++);
```

```
    out.println("<tr><td><input type=radio name=BROKERID value=\"" + brokerId + "\">");
            out.println("</td><td>");
            out.println("Broker ID:");
            out.println("</td><td>");
            out.println(brokerId);
            out.println("</td><tr><td></td><td>");
            out.println("Broker Name:");
            out.println("</td><td>");
            out.println(fname + " " + lname);
            out.println("</td></tr>");
          }
          bInfo.reset();
          session.invalidate();
          out.println("</table>");
          out.println("<p><input type=\"submit\" value=\"Make an appointment\"></p>");
          out.println("</form>");
          out.println("</body></html>");
          out.close();
 }

 public String getServletInfo()
 {                return "A servlet that perform property search on the sweethome database";
 }
}
```

crServlet.java

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.*;
import java.beans.*;
import java.io.*;
import java.util.*;
import java.text.*;

public class crServlet extends HttpServlet
{ private accessBean aB = null;
  public PropertyChangeAdapter adapter = new PropertyChangeAdapter();
  public boolean queryStatus = false;
  //private boolean flag = true;
  private String userid = "root";
  private String passwd = "123456";

  public void doPost(HttpServletRequest req, HttpServletResponse res)
     throws ServletException, IOException
  { res.setContentType("text/html");
    ServletOutputStream out = res.getOutputStream();
    out.println("<html>");
    out.println("<head><title>Customer Registration Page</title></head>");
    out.println("<body>");
```

```
boolean flag = true;
String ssn = req.getParameter("SSN1") + req.getParameter("SSN2") +
             req.getParameter("SSN3");
String fName = req.getParameter("FIRST");
String mName = req.getParameter("MID");
String lName = req.getParameter("LAST");
String street = req.getParameter("STREET");
String city = req.getParameter("CITY");
String state = req.getParameter("STATE");
String zip = req.getParameter("ZIP");
String home = req.getParameter("HOME1") + req.getParameter("HOME2") +
              req.getParameter("HOME3");
String work = req.getParameter("WORK1") + req.getParameter("WORK2") +
              req.getParameter("WORK3");
String dateFormat = "yyyy-MM-dd";
SimpleDateFormat formatter = new SimpleDateFormat(dateFormat);
java.util.Date dt = new java.util.Date(System.currentTimeMillis());
String age = formatter.format(dt);

if(ssn.length() == 0)
{  out.println("<br>Error !  SSN is required");
   flag = false;
}
else if(ssn.length() != 9)
{  out.println("<br>Error !  SSN is invalid");
   flag = false;
}
if(fName.length() == 0)
{  out.println("<br>Error !  First Name is required");
   flag = false;
}
if(lName.length() == 0)
{  out.println("<br>Error !  Last Name is required");
   flag = false;
}
if(street.length() == 0)
{  out.println("<br>Error !  Street is required");
   flag = false;
}
if(city.length() == 0)
{  out.println("<br>Error !  City is required");
   flag = false;
}
if(state.length() == 0)
{  out.println("<br>Error !  State is required");
   flag = false;
}
else if(state.length() != 2)
{  out.println("<br>Error !  State is invalid");
   flag = false;
}
if(zip.length() == 0)
```

```
{  out.println("<br>Error !  Zip Code is required");
   flag = false;
}
else if(zip.length() != 5)
{  out.println("<br>Error !  Zip Code is invalid");
   flag = false;
}
if(home.length() == 0)
{  out.println("<br>Error !  Home Phone is required");
   flag = false;
}
else if(home.length() != 10)
{  out.println("<br>Error !  Home Phone is invalid");
   flag = false;
}
if(work.length() != 0 && work.length() != 10)
{  out.println("<br>Error !  Work Phone is invalid");
   flag = false;
}

if(flag)
{  String sql = "SELECT ssn FROM customer WHERE ssn = '" + ssn + "'";
   aB = new accessBean("sweethome", userid, passwd, sql);
   aB.addPropertyChangeListener(adapter);
   aB.execQuery();

   if(aB.getResultArray(0) != null)
   {  out.println("<p><b>ERROR !!!  SSN has been used !!!</b>");
      out.println("<br><br>");
      flag = false;
   }
}

if(flag)
{  String sql = "INSERT INTO customer set SSN='" + ssn + "',LName='" + lName +
                   "',MName='" + mName + "',FName='" + fName + "',HomeNo='" + home +
              "',WorkNo='" + work + "',Street='" + street + "',City='" + city + "',State='" + state +
              "',Code='" + zip + "',Regdate='" + age + "' ";
   aB = new accessBean("sweethome", userid, passwd, sql);
   aB.addPropertyChangeListener(adapter);
   aB.execUpdate();

   if(!getQueryStatus())
   {  out.println("<p><b>Error ! " + getErrorMsg() + "</b>");
      flag = false;
   }
}

if(flag)
{  out.println("<p><br><br><b>Congratulations</b>");
   out.println("<p><b>Your Registration has confirmed !</b>");
   out.println("<p><b>Please use your SSN as your ID to browse this website !</b>");
```

```
    }
    out.println("<p><br><br><hr>");
   /* if(!flag)
    { out.println("<p><br><a href='../cr.html'>GO BACK !</a><br>");
    }*/
    out.println("</body></html>");
  }

  public String getServletInfo()
  { return "A Servlet that performs Sweet Home Inc. database access";
  }
  //    Get the error message

  public synchronized String getErrorMsg()
  { return aB.getErrorMsg();
  }
  public synchronized void setQueryStatus(boolean b)
  { queryStatus = b;
  }
  public synchronized boolean getQueryStatus()
  { return queryStatus;
  }
  class PropertyChangeAdapter implements PropertyChangeListener
  { public void propertyChange(PropertyChangeEvent e)
    { String status = new String(e.getNewValue().toString());
        if(status.equals("true"))
          { if(aB.getErrorMsg() != null)
          setQueryStatus(false);
          else
                setQueryStatus(true);
          }
    }
  }
}
```

appServlet.java

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.*;
import java.beans.*;
import java.io.*;
import java.util.*;
import java.text.*;

public class appServlet extends HttpServlet
{ public void doPost(HttpServletRequest req, HttpServletResponse res)
     throws ServletException, IOException
  { res.setContentType("text/html");
    ServletOutputStream out = res.getOutputStream();
    out.println("<html>");
```

```
        out.println("<head><title>MAKE APPOINTMENT</title></head>");
        out.println("<body>");
        String bid = req.getParameter("BROKERID");
        out.println("<table><tr><td><IMG SRC='../../kevin-p/images/handshake.jpg' width=75
                                                                    height=100></td>");
            out.println("<td valign=bottom><H2><B><U>SET
    UP<P>APPOINTMENT</U></B></H2></td></tr>");
            out.println("</table><br>");
            out.println("<form method='POST' action='/servlet/makeAppServlet'>");
        out.println("<table>");
        out.println("<tr><td width=100>Broker Id:</td><td width=150>" +
              "<input type='text' name='BID' size='20' value='" + bid + "'></td></tr>");
            out.println("<tr><td>Customer id:</td><td><input type='text' name='CID' size='20'>" +
                  "</td></tr><tr><td>Date:</td><td>mm  <SELECT NAME='MONTH'>");
        out.println("<OPTION VALUE='1'>Jan</OPTION>");
        out.println("<OPTION VALUE='2'>Feb</OPTION>");
        out.println("<OPTION VALUE='3'>Mar</OPTION>");
        out.println("<OPTION VALUE='4'>Apr</OPTION>");
        out.println("<OPTION VALUE='5'>May</OPTION>");
        out.println("<OPTION VALUE='6'>Jun</OPTION>");
        out.println("<OPTION VALUE='7'>Jul</OPTION>");
        out.println("<OPTION VALUE='8'>Aug</OPTION>");
        out.println("<OPTION VALUE='9'>Sep</OPTION>");
        out.println("<OPTION VALUE='10'>Oct</OPTION>");
        out.println("<OPTION VALUE='11'>Nov</OPTION>");
        out.println("<OPTION VALUE='12'>Dec</OPTION>");
        out.println("</SELECT>");
        out.println("dd  <SELECT NAME='DATE'>");
        out.println("<OPTION VALUE='1'>1</OPTION>");
        out.println("<OPTION VALUE='2'>2</OPTION>");
        out.println("<OPTION VALUE='3'>3</OPTION>");
        out.println("<OPTION VALUE='4'>4</OPTION>");
        out.println("<OPTION VALUE='5'>5</OPTION>");
        out.println("<OPTION VALUE='6'>6</OPTION>");
        out.println("<OPTION VALUE='7'>7</OPTION>");
        out.println("<OPTION VALUE='8'>8</OPTION>");
        out.println("<OPTION VALUE='9'>9</OPTION>");
        out.println("<OPTION VALUE='10'>10</OPTION>");
        out.println("<OPTION VALUE='11'>11</OPTION>");
        out.println("<OPTION VALUE='12'>12</OPTION>");
        out.println("<OPTION VALUE='13'>13</OPTION>");
        out.println("<OPTION VALUE='14'>14</OPTION>");
        out.println("<OPTION VALUE='15'>15</OPTION>");
        out.println("<OPTION VALUE='16'>16</OPTION>");
        out.println("<OPTION VALUE='17'>17</OPTION>");
        out.println("<OPTION VALUE='18'>18</OPTION>");
        out.println("<OPTION VALUE='19'>19</OPTION>");
        out.println("<OPTION VALUE='20'>20</OPTION>");
        out.println("<OPTION VALUE='21'>21</OPTION>");
        out.println("<OPTION VALUE='22'>22</OPTION>");
        out.println("<OPTION VALUE='23'>23</OPTION>");
        out.println("<OPTION VALUE='24'>24</OPTION>");
```

```
    out.println("<OPTION VALUE='25'>25</OPTION>");
    out.println("<OPTION VALUE='26'>26</OPTION>");
    out.println("<OPTION VALUE='27'>27</OPTION>");
    out.println("<OPTION VALUE='28'>28</OPTION>");
    out.println("<OPTION VALUE='29'>29</OPTION>");
    out.println("<OPTION VALUE='30'>30</OPTION>");
    out.println("<OPTION VALUE='31'>31</OPTION>");
    out.println("</SELECT>");
    out.println("yy <SELECT NAME='YEAR'>");
    out.println("<OPTION VALUE='2002'>2002</OPTION>");
    out.println("<OPTION VALUE='2003'>2003</OPTION>");
    out.println("</SELECT>");
    out.println("</td></tr>");
    out.println("<tr><td valign=top>Time:</td>");
          out.println("<td><input type='radio' value='10:00' name='TIME' CHECKED>10:00
AM" +  "<BR><input type='radio' value='14:00' name='TIME'>02:00 PM</td></tr>");
    out.println("<tr><td valign=top>Purpose:</td>");
    out.println("<td><input type='radio' name='PURPOSE' value='BUY'
                                                        CHECKED>Buy<BR>");
    out.println("<input type='radio' value='SELL' name='PURPOSE'>Sell<BR>");
    out.println("<input type='radio' value='OTHER' name='PURPOSE'>Other");
    out.println("</td></tr>");
    out.println("</table>");
    out.println("<p><input type='submit' value='Submit'> <input type='reset' value='Reset'>");
    out.println("</form>");
    out.println("<p>[<a href='../../kevin-p/forgetid.html'>Forget your ID?</a>]" +
          "[<a href='broker.html'>View broker information</a>]");
    out.println("</body>");
    out.println("</html>");
  }

  public String getServletInfo()
  { return "Make Appointment page with Broker ID ready";
  }
}
```

makeAppServlet.java

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.*;
import java.beans.*;
import java.io.*;
import java.util.*;
import java.text.*;

public class makeAppServlet extends HttpServlet
{ private accessBean aB = null;
  public PropertyChangeAdapter adapter = new PropertyChangeAdapter();
  public boolean queryStatus = false;
  private boolean flag = true;
```

```
private String userid = "root";
private String passwd = "123456";
String dateFormat = "yyyy-MM-dd";
SimpleDateFormat formatter = new SimpleDateFormat(dateFormat);
java.util.Date dt = new java.util.Date(System.currentTimeMillis());
String curr = formatter.format(dt);
String currD = curr.substring(8,10);
String currM = curr.substring(5,7);
String currY = curr.substring(0,4);

public void doPost(HttpServletRequest req, HttpServletResponse res)
   throws ServletException, IOException
{  res.setContentType("text/html");
  ServletOutputStream out = res.getOutputStream();
  out.println("<html>");
  out.println("<head><title>Make Appointment</title></head>");
  out.println("<body>");
  String bid = req.getParameter("BID");
  String cid = req.getParameter("CID");
  String month = req.getParameter("MONTH");
  String date = req.getParameter("DATE");
  String year = req.getParameter("YEAR");
  String time = req.getParameter("TIME");
  String purpose = req.getParameter("PURPOSE");
  String appDate = year + "-" + month + "-" + date;

  if(date.length() == 1) date = "0" + date;
  if(month.length() == 1) month = "0" + month;

  if(bid.length() == 0)
  {  out.println("<br><b>Error!  Broker ID is REQUIRED</b>");
    flag = false;
  }
  else if(bid.length() != 5)
  {  out.println("<br><b>Error!  Broker ID is INVALID</b>");
    flag = false;
  }
  if(cid.length() == 0)
  {  out.println("<br><b>Error!  Customer ID is REQUIRED</b>");
    flag = false;
  }
  else if(cid.length() != 9)
  {  out.println("<br><b>Error!  Customer ID is INVALID</b>");
    flag = false;
  }
 // Date Checking
  if(date.equals("31"))
  {  if(month.equals("02") || month.equals("04") || month.equals("06") ||
      month.equals("09") || month.equals("11"))
    {  out.println("<p><b>Error!  Date is invalid...</b>");
      flag = false;
        }
```

```
    }
    else if(date.equals("30"))
    { if(month.equals("02"))
      { out.println("<p><b>Error! Date is invalid...</b>");
        flag = false;
            }
    }
    else if(date.equals("29") && month.equals("02") && !year.equals("2000"))
    { out.println("<p><b>Error! Date is invalid...</b>");
      flag = false;
    }

    if(flag)
    { if(year.equals(currY) && month.equals(currM) && date.equals(currD))
      { out.println("<p><b>Sorry, you cannot make an appointment for today</b>");
        out.println("<p><b>Please set it up at least 1 day in advance</b>");
        flag = false;
      }
      else if( (year.compareTo(currY) < 0) || (year.compareTo(currY) == 0 &&
month.compareTo(currM) < 0)
            || (year.compareTo(currY) == 0 &&  month.compareTo(currM) ==0  &&
date.compareTo(currD) < 0))
        { out.println("<p><b>Error! Time you've chosen was in the past...</b>");
          out.println("<p><b>Please choose another time...</b>)");
          flag = false;
        }
    }

    if(flag)
    { String sql = "SELECT brokerid FROM broker WHERE brokerid = '" + bid + "'";
      aB = new accessBean("sweethome", userid, passwd, sql);
      aB.addPropertyChangeListener(adapter);
      aB.execQuery();

      if(aB.getResultArray(0) == null)
      { out.println("<p><b>We're sorry,</b>");
        out.println("<p><b>The Broker ID doesn't exist...</b>");
        out.println("<br><br>");
        flag = false;
      }

      if(!getQueryStatus())
      { out.println("ERROR - " + getErrorMsg());
      }

      sql = "SELECT ssn FROM customer WHERE ssn = '" + cid + "'";
      aB = new accessBean("sweethome", userid, passwd, sql);
      aB.addPropertyChangeListener(adapter);
      aB.execQuery();

      if(aB.getResultArray(0) == null)
      { out.println("<p><b>We're sorry,</b>");
```

```
      out.println("<p><b>Your Customer ID doesn't exist...</b>");
      out.println("<br><br>");
      flag = false;
    }
    if(!getQueryStatus())
    { out.println("ERROR - " + getErrorMsg());
    }
  }

  if(flag)
  { String sql = "SELECT appdate, apptime FROM appointment WHERE brokerid = '" +
              bid + "' AND customerssn = '" + cid + "'";
    aB = new accessBean("sweethome", userid, passwd, sql);
    aB.addPropertyChangeListener(adapter);
    aB.execQuery();

    if(aB.getResultArray(0) != null &&
      aB.getResultArray(0).equals(year + "-" + month + "-" + date) &&
      aB.getResultArray(1).equals(time))
    { out.println("<p><b>We're sorry, Broker is not available at that time!</b>");
      out.println("<p><b>Please choose some other time...</b>");
      flag = false;
    }
  }

  if(flag)
  { String sql = "INSERT INTO appointment VALUES('" +
              bid + "','" + cid + "','" + appDate + "','" +
              time + "','" + purpose + "')";
    aB = new accessBean("sweethome", userid, passwd, sql);
    aB.addPropertyChangeListener(adapter);
    aB.execUpdate();

    if(!getQueryStatus())
    { out.println("<p><b>Error ! " + getErrorMsg() + "</b>");
      flag = false;
    }
  }

  if(flag)
  { out.println("<p><br><br><b>Congratulations</b>");
    out.println("<p><b>Your Appointment has been confirmed !</b>");
  }

  out.println("<p><br><br><hr>");
  if(!flag)
  { out.println("<p><br><a href=\"/kevin-p/app.html\">GO BACK !</a><br>");
  }
  flag = true;
  out.println("</body></html>");
}
```

```
  public String getServletInfo()
  { return "A Servlet that performs Sweet Home Inc. Appointment page";
  }
  //    Get the error message

  public synchronized String getErrorMsg()
  { return aB.getErrorMsg();
  }

  public synchronized void setQueryStatus(boolean b)
  { queryStatus = b;
  }

  public synchronized boolean getQueryStatus()
  { return queryStatus;
  }

  class PropertyChangeAdapter implements PropertyChangeListener
  { public void propertyChange(PropertyChangeEvent e)
    { String status = new String(e.getNewValue().toString());

      if(status.equals("true"))
      { if(aB.getErrorMsg() != null)
          setQueryStatus(false);
        else
          setQueryStatus(true);
      }
    }
  }
}
```

forgetidServlet.java

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.*;
import java.beans.*;
import java.io.*;
import java.util.*;

public class forgetidServlet extends HttpServlet
{ private accessBean aB = null;
  public PropertyChangeAdapter adapter = new PropertyChangeAdapter();
  public boolean queryStatus = false;
  private String userid = "root";
  private String passwd = "123456";

  public void doPost(HttpServletRequest req, HttpServletResponse res)
     throws ServletException, IOException
  { res.setContentType("text/html");
    ServletOutputStream out = res.getOutputStream();
```

```
out.println("<html>");
out.println("<head><title>Customer Registration Page</title></head>");
out.println("<body>");
String ssn = req.getParameter("SSN1") + req.getParameter("SSN2") +
             req.getParameter("SSN3");
String zip = req.getParameter("ZIP");
boolean flag = true;

if(ssn.length() == 0)
{  out.println("<br>Error !  SSN is required");
   flag = false;
}
else if(ssn.length() != 9)
{  out.println("<br>Error !  SSN is invalid");
   flag = false;
}
if(zip.length() == 0)
{  out.println("<br>Error !  Zip Code is required");
   flag = false;
}
else if(zip.length() != 5)
{  out.println("<br>Error !  Zip Code is invalid");
   flag = false;
}

if(flag)
{  String sql = "SELECT ssn, code, fname FROM customer WHERE ssn = '" + ssn + "'";
   aB = new accessBean("sweethome", userid, passwd, sql);
   aB.addPropertyChangeListener(adapter);
   aB.execQuery();
   if(aB.getResultArray(0) == null)
   {  out.println("<p>We're sorry ....");
      out.println("<p><b>Your SSN is not registered</b>");
      out.println("<p><br>Please go to the <a href='kevin-p/cr.html'>Customer Registration
                                                                    page.</a><br>");
      flag = false;
   }
   else
   {  if(aB.getResultArray(1).equals(zip))
      {  out.println("<p><br>Welcome, <b>" + aB.getResultArray(2) + "</b>...");
         out.println("<p>You are a registered customer !");
         out.println("<p>Please use your <b>SSN</b> as your <b>user id</b> and");
         out.println("<br>your <b>zip code</b> as your <b>password</b>");
      }
      else
      {  out.println("<p>Sorry, you're a registered customer,");
         out.println("<p>but your zipcode doesn't match !");
         out.println("<p>Please contact our customer service !");
         out.println("<p><br><br><a href='../sweethome/forgetid.html'>Try Again</a>");
      }
   }
}
```

```
    out.println("<p><br><br><hr>");
    if(!flag)
    { out.println("<p><br><a href='../sweethome/forgetid.html'>GO BACK !</a><br>");
    }
    out.println("</body></html>");
  }

  public String getServletInfo()
  { return "When a customer forget their id or password";
  }
 //   Get the error message
  public synchronized String getErrorMsg()
  { return aB.getErrorMsg();
  }

  public synchronized void setQueryStatus(boolean b)
  { queryStatus = b;
  }

  public synchronized boolean getQueryStatus()
  { return queryStatus;
  }

  class PropertyChangeAdapter implements PropertyChangeListener
  { public void propertyChange(PropertyChangeEvent e)
    { String status = new String(e.getNewValue().toString());

      if(status.equals("true"))
      { if(aB.getErrorMsg() != null)
          setQueryStatus(false);
        else
          setQueryStatus(true);
      }
    }
  }
}
```

REFERENCES

[1] Marty Hall and Larry Brown, "Core Web programming", Prentice Hall, 2001.

[2] Jose Annunziato and Stephanie Fesler Kaminaris, "JavaServer Pages in 24 hours", Sams Publishing, 2001.

[3] IEEE Recommended Practice for Software Requirements Specifications (IEEE Std 830-1993).

[4] Java Tutorial, http://java.sun.com/docs/books/tutorial/

[5] Martin Fowler, "*UML distilled: a brief guide to the standard object modeling language*, 2nd ed." Addison Wesley Longman Inc., 1999.

[6] Java API Specifications, http://java.sun.com/products/jdk/1.3/docs/api/index.html

[7] Tyler Jewell and etc., "Java Server Programming J2EE 1.3 Edition" Wrox Press Ltd., 2001.

[8] The Jakarta Project, http://jakarta.apache.org/tomcat/

[9] MySQL, http://www.mysql.com/